AD-A085 289   COMMAND AND CONTROL TECHNICAL CENTER WASHINGTON DC   F/G 15/7
THE CCTC QUICK-REACTING GENERAL WAR GAMING SYSTEM (QUICK). PROG--ETC(U)
APR 80

UNCLASSIFIED   NL

1 OF 3
AD
A085289

(12)

DEFENSE COMMUNICATIONS AGENCY
COMMAND AND CONTROL
TECHINCAL CENTER
WASHINGTON, D.C. 20301

LEVEL III

4072 821

IN REPLY REFER TO: C314

30 April 1980

TO: RECIPIENTS

SUBJECT: Change 2 to Program Maintenance Manual CSM MM 9-77, Volume III, Weapon Allocation Subsystem

1. Insert the enclosed change pages and destroy the replaced pages according to applicable security regulations.

2. A list of Effective Pages to verify the accuracy of this manual is enclosed. This list should be inserted before the title page.

3. When this change has been posted, make an entry in the Record of Changes.

FOR THE DIRECTOR:

129 Enclosures
Change 2 Pages

J. DOUGLAS POTTER
Assistant to the Director
for Administration

The CCTC Quick Reacting General War Gaming System (QUICK). Program Maintenance Manual. Volume III. Weapon Allocation Subsystem. Change 2.

409658

EFFECTIVE PAGES - APRIL 1980

This list is used to verify the accuracy of CSM MM 9-77, Volume III after change 2 pages have been inserted. Original pages are indicated by the letter O, change 1 pages by the numeral 1, and change 2 pages by the numeral 2.

Page No.	Change No.	Page No.	Change No.
Title Page	0	69	2
ii-xi	2	70-71	0
1	0	72	2
2	2	73-75	0
3-7	1	76	2
8	2	77-83	0
9	1	84-84.2	2
10-14	2	85-87.2	2
15-21	1	88-89.2	2
22	2	90-94	2
23-24	1	95-98	0
25-26	2	99	2
27-28	1	100-103	0
29	2	104	1
30-37	1	105	2
38	2	106-108	0
39-41	1	109	1
42	2	110	2
43	1	111-112	0
44-46	2	123-126	1
47-53	1	126.1-126.2	1
54	2	127-152	0
55-57	1	153	2
58	2	154	0
59-60	1	155	1
60.1-60.14	1	156-165	0
60.15-60.16	2	166	1
60.17	1	167	2
60.18	2	168	0
60.19-60.26	1	169	2
60.27	2	170	0
60.28-60.34	1	171	2
61-62.4	2	172-174	0
63	2	175-178.2	2
64	1	179-181	2
65-68	0	182-183	0
		184-185	2

Page No.	Change No.	Page No.	Change No.
186-189	0	291-292	2
190	2	293	0
191-197	0	294	2
198-202	2	295-302	0
203-206	0	303	2
207	2	304-320	0
208	0	321	2
209	2	322-346	0
210	0	347	2
211-212	2	348-356	0
213	0	357	2
214	2	358-364	0
215-216	0	365	2
217	2	366-367	1
218	0	368	2
219	2	369	1
220-226	0	370	0
227-231	2	371	1
232-237	0	372-374	0
238	2	375	2
239-240	0	376-380	0
240.1-240.12	2	381-382	2
241-243	2	383-384	0
244	0	385-390	2
245-251	2	391-398	0
252-253	0	399	2
254	2	400	0
255-257	0	401-403	2
258	2	404	0
259-260	0	405-405.2	2
261	2	406-409	0
262	0	410	1
263-264	2	411	0
265-267	0	412-418.2	2
268	2	419-421.2	2
269	0	422-422.3	1
270	2	422.4-422.5	2
271-272	0	422.6-422.10	1
273	2	423-430	0
274	0	431	2
275-278	2	432-452	0
279-280	0	453-454	2
281-282	2	455-465	0
283-284	0	466	2
285-286	2	467-470	0
287-290	0	471-472	2

Page No.	Change No.
473-498	0
499	2
500-512	0
513	2
514-516	0

ACKNOWLEDGMENT

This document was prepared under the direction of the Chief for Military Studies and Analyses, CCTC, in response to a requirement of the Studies, Analysis, and Gaming Agency, Organization of the Joint Chiefs of Staff. Technical support was provided by System Sciences, Incorporated under Contract Number DCA 100-75-C-0019. Change set one was prepared under Contract Number DCA 100-78-C-0035. Computer Sciences Corporation prepared change set two under Contract Number DCA 100-78-C-0042.

CONTENTS

Section Page

Section Page

Section Page

ILLUSTRATIONS

Figure Page

TABLES

ABSTRACT

The computerized Quick-Reacting General War Gaming System (QUICK) will accept input data, automatically generate global strategic nuclear war plans, provide statistical output summaries and produce input tapes to simulator subsystems external to QUICK. QUICK has been programmed in FORTRAN for use on the CCTC HIS 6000 computer system.

The QUICK Program Maintenance Manual consists of four volumes: Volume I, Data Management Subsystem; Volume II, Weapon/Target Identification Subsystem; Volume III, Weapon Allocation Subsystem; Volume IV, Sortie Generation Subsystem. The Program Maintenance Manual complements the other QUICK Computer System Manuals to facilitate maintenance of the war gaming system. This volume, Volume III, provides the programmer/analyst with a technical description of the purpose, functions, general procedures, and programming techniques applicable to the programs and subroutines of the Weapon Allocation subsystem. Companion documents are:

a. USERS MANUAL
Computer System Manual UM 9-77, Volume I
Computer System Manual UM 9-77, Volume II
Computer System Manual UM 9-77, Volume III
Computer System Manual UM 9-77, Volume IV
Provides detailed instructions for applications of the system.

b. TECHNICAL MEMORANDUM
Technical Memorandum TM 153-77
Provides a nontechnical description of the system for senior management personnel.

SECTION 1. GENERAL

1.1 Purpose

This volume of the QUICK Program Maintenance Manual describes the modules which are part of the QUICK Weapon/Allocation subsystem, detailing the modules, subroutines, and functions which it comprises. The information contained herein is presented on a module-by-module basis. The module-by-module discussions are structured so that a maintenance programmer can understand the program functions and programming techniques. The computer subjects are structured to inform the maintenance programmer of overall system programming techniques and conventions.

1.2 General Description

The Weapon Allocation subsystem operates using the integrated data base as defined by all modules up to PLANSET of the Weapon/Target Identification subsystem and produces a plan using the weapon resources specified to maximize the expected target value destroyed. The subsystem consists of modules PREPALOC, ALOC, EVALALOC, and ALOCOUT, as shown in figure 1. Figure 2 shows the relationship of the Weapon Allocation subsystem to other QUICK subsystems in terms of procedural and information flow.

The modules and supporting subroutines of this subsystem are used to define information for use in later processes and allocate given weapons to targets to optimize expected value destroyed. The integrated data base is updated as each module is exercised in sequence. The final output provides proper inputs necessary to execute the Sortie Generation subsystem.

The first module, PREPALOC, precomputes much of the information required by later processors. In addition, it provides capabilities for planning factor modification and fixed weapon assignment specification.

The next module, ALOC, performs the allocation of weapons to targets. Using a generalized Lagrange multiplier method, an optimal allocation is generated subject to several forms of user-input allocation constraints. These constraints include specification of minimum and maximum desired damage levels, restriction of weapons to specified subsets of the target system, and specification of weapons allocated to specific targets by the user. Within these constraints, the program generates the allocation which maximizes the expected value destroyed in the target system. Module ALOC is also referred to as the allocator.

The main function of module EVALALOC is to provide a summary of the allocation produced in module ALOC and to calculate an expected-value estimate of its results. In addition, the module has the capability of evaluating the effect upon the results of variations in input values for weapon and target parameters. Module EVALALOC may be run either before module ALOCOUT or after module PLANOUT.

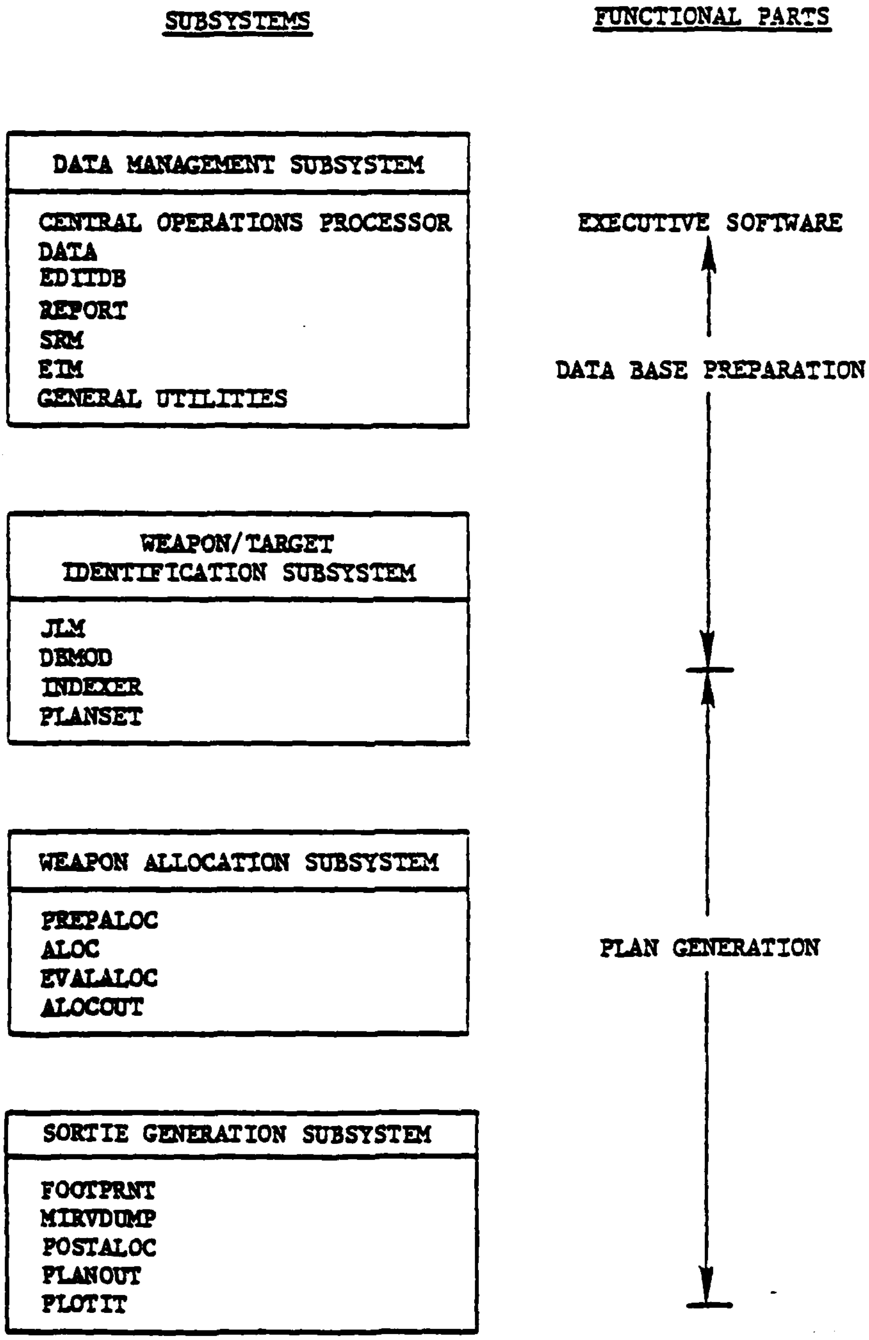


Figure 1. Major Subsystems of the QUICK System

2.5.3 Subroutine SETRD. This subroutine is called in any execution which contains a SETTING clause. It performs changes to general gaming parameters and weapon group height of bursts. Changes to target parameters are stored on an external data file (25) for subroutine CHGBAS.

2.5.4 Subroutine CHGBAS. This subroutine is called in any execution where subroutine SETRD has stored one or more change requests on file 25. These requests are carried out for the target base according to a preset priority scheme.

2.5.5 Subroutine FIXWEP. This subroutine is called in any execution with one or more FIX clauses. It creates fixed assignment records (ASSIGN) for use by the allocation module.

2.5.6 Subroutine PRNPRP. This subroutine performs all standard and optional prints. It is called at the end of every run and produces those optional prints requested and any standard prints called for by the input.

2.6 PREPALOC Internal Common Blocks

All common blocks used internally by PREPALOC are given in table 1. For definition of common blocks that communicate with the COP, see Program Maintenance Manual, Volume I.

Table 1. Module PREPALOC Common Blocks
(Part 1 of 2)

COMMON	VARIABLE OR ARRAY	DESCRIPTION
CRLENGTH	CRLENGTH(30)	Total length of a penetration corridor; indexed by corridor number
DISTEF	DISTEF(50)	Total length of a depenetration corridor; indexed by corridor number
	DISTEG(50)	Total length of a depenetration corridor plus distance to nearest recovery base; indexed by corridor number
	DRECOV(50,4)	DESIG of recovery base associated with depenetration corridor; indexed by corridor number, up to four bases
	DISTNA(50,4)	Distance from depenetration corridor to recovery base; indexed by corridor number, up to four bases
	INDXA(50,4)	Ordinal of bases ordered on distance from depenetration corridor; indexed by corridor number, up to four bases
FIXOFF	OFFSW	Switch to indicate if fix is to be offset (.true.) or not (.false.)
GEOREF	IPNRF	IDS reference code of penetration corridor header
	IDPRF	IDS reference code of depenetration corridor header
IFXREQ	IFXREQ(250)	Fixed assignment requests; indexed by group
	IFXHON(250)	Fixed assignment requests honored; indexed by group
IFXSW	FIXSW	Fixed assignment switch: true if there are assignments in this run, false otherwise
NTBFLG	NTBFLG(9)	Alphanumeric value to indicate source of values in general gaming parameters. May be DEFAUL for default, UNCHNG for changed during this run

Table 1. (Part 2 of 2)

ASSOCIATED	VARIABLE	
COMMON	OR ARRAY	DESCRIPTION
RFPOINTS	RFLAT(20)	Latitutude of a refuel point
	RFLONG(20)	Longitude of a refuel point
	MREGN	Number of refuel points
SETCOM	NTBCH	General gaming parameter switch: true if changes were made, false otherwise
	DBCH	Target parameter switch: true if changes were made, false otherwise
	NDBCH	Number of target parameter change requests written onto file 25 by SETRD
	RECALC	RECALC mode switch: true if RECALC mode active, false otherwise
TSTUFF		Communicates with TOFM
	XTOFMIN	Minimum time of flight
	XCMISS	Missile flight parameter
	XRNGMIN	Minimum range
	XRANGE	Missile maximum range
WEPCOM	IWRCD(250)	Weapon group record IDS reference code; indexed on group
	IWTYP(250)	Weapon group type index
	IWCLS(250)	Weapon group class indicator (1=missile, 2=bomber, 3=salvoed missile)
	NMWTYP(120)	Weapon type name; indexed on type
	IWMIRV(250)	MIRV indicator (0=nonMIRV, 1=MIRV)
	NASM(250)	Number of ASMs in group payload
	NMGRP	Number of groups

2.7 Subroutine ENTMOD

PURPOSE: To control overall flow of processing

ENTRY POINTS: ENTMOD (first subroutine called when overlay link PREP is executed)

FORMAL PARAMETERS: None

COMMON BLOCKS: C15, C30, ERRCOM, FIXOFF, IFXREQ, IFXSW, NTBFLG, SETCOM

SUBROUTINES CALLED: CHGBAS, FIXWEP, GEOPREP, HDFND, INSGET, MODFY, PRNPRP, RETRV, SETRD, WEPPREP

CALLED BY: COP

Method:

The basic method employed by the driver routine of PREPALOC is to check for particular input clauses in a set order and carry out functions based on their presence. As a preliminary step, the IDS error code "R04" is set as acceptable so that the retrieval of nonexistent DESIG values will not cause termination and the number table (NUMTBL) is retrieved. The input is then scanned for a RECALC clause, if it exists the general gaming parameters (INITSTRK, CORMSL, etc.) are set to their default values and GEOPREP and WEPPREP are called.

The subroutine now looks for any SETTING clauses. Subroutine SETRD is called for each occurrence. If any SETTING clause called for changes to the target parameters (VALUE, MINKILL, MAXKILL or IDHOB), CHGBAS is also called. Then, the input is scanned for FIX clauses and FIXWEP called at each occurrence. Finally, PRNPRP is called to produce any standard or optional print.

Subroutine ENTMOD is illustrated in figure 3.

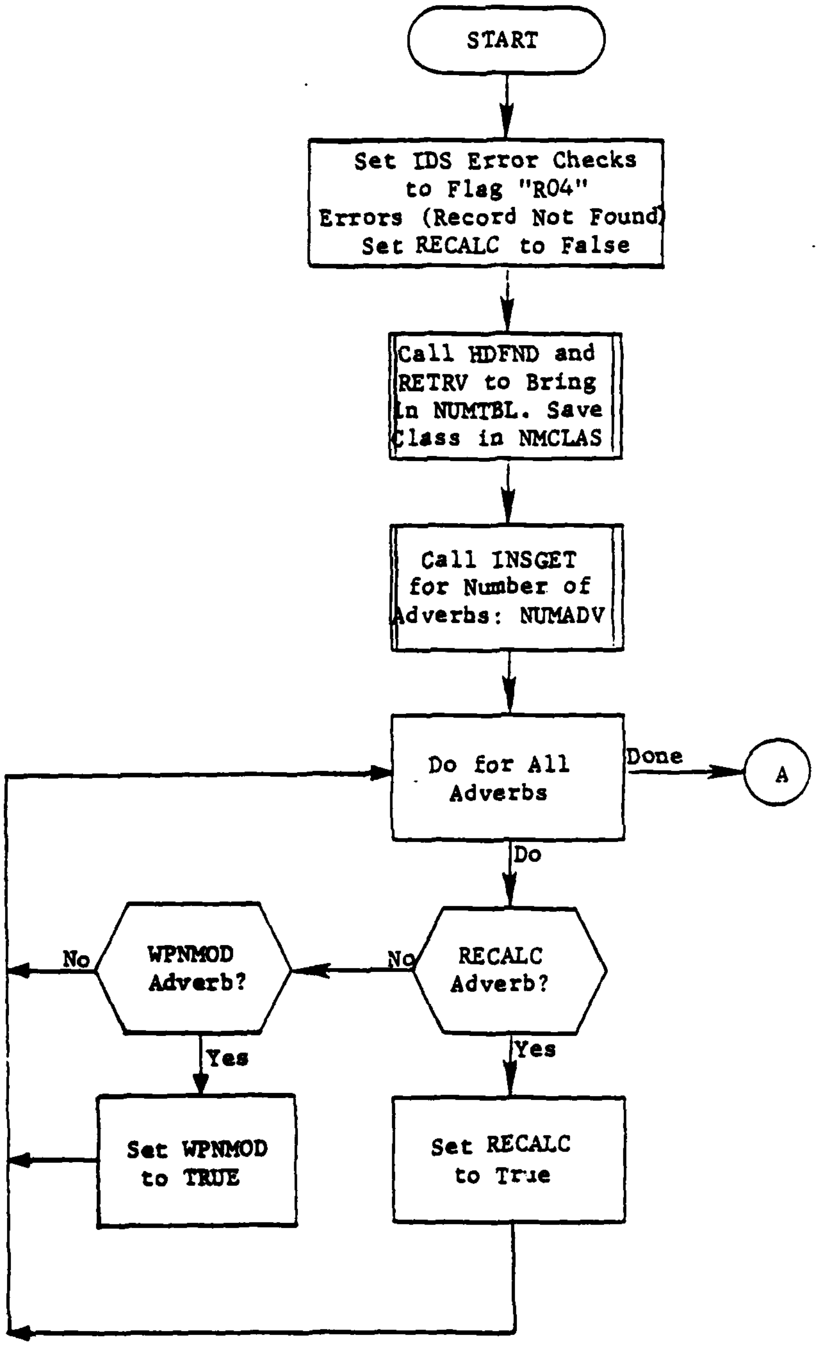

START
Set IDS Error Checks to Flag "R04" Errors (Record Not Found) Set RECALC to False
Call HDFND and RETRV to Bring in NUMTBL. Save Class in NMCLAS
Call INSGET for Number of Adverbs: NUMADV
Do for All Adverbs
Done
A
Do
RECALC Adverb?
No
Yes
WPNMOD Adverb?
No
Yes
Set WPNMOD to TRUE
Set RECALC to True

Figure 3. Subroutine ENTMOD (PREPALOC) (Part 1 of 4)

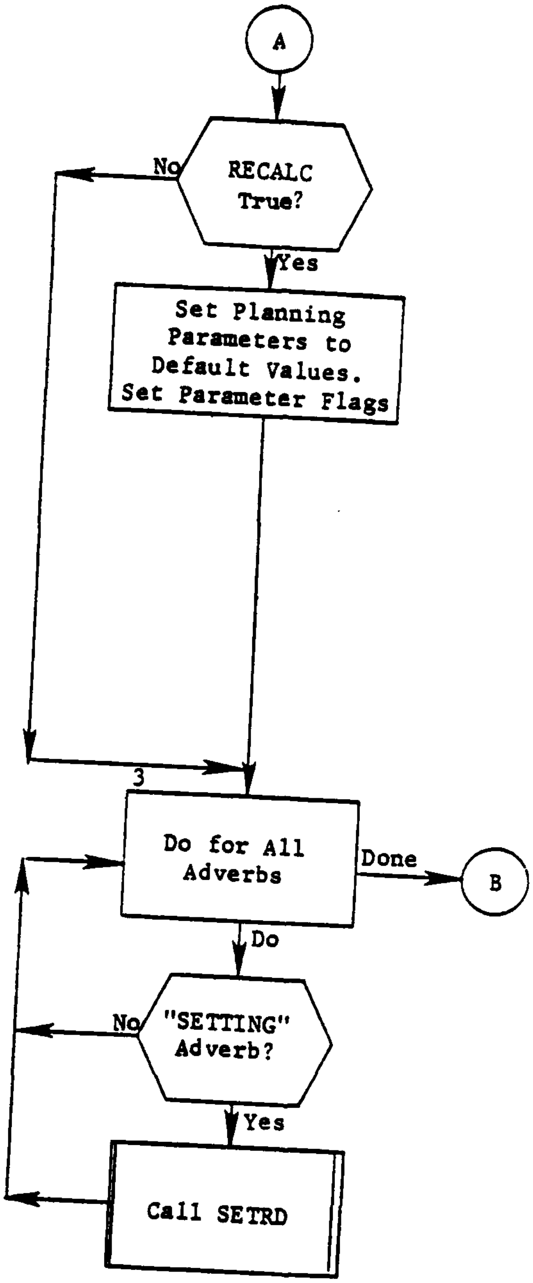


Figure 3. (Part 2 of 4)

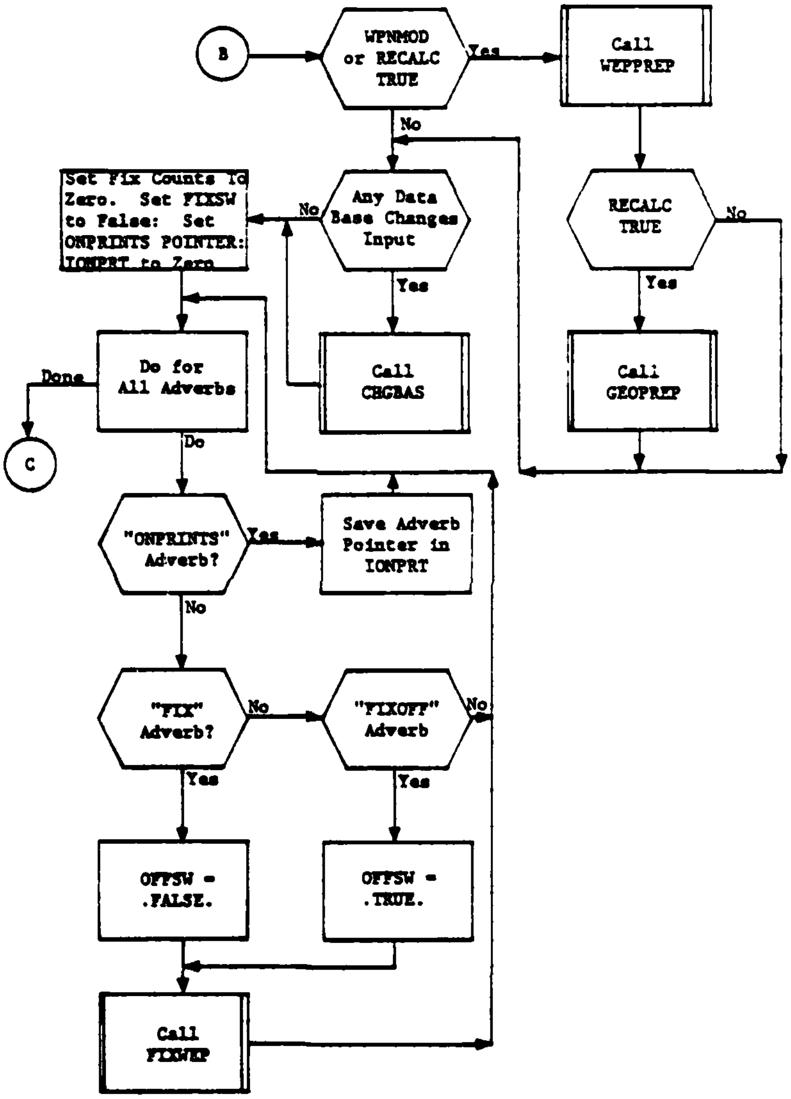


Figure 3. (Part 3 of 4)

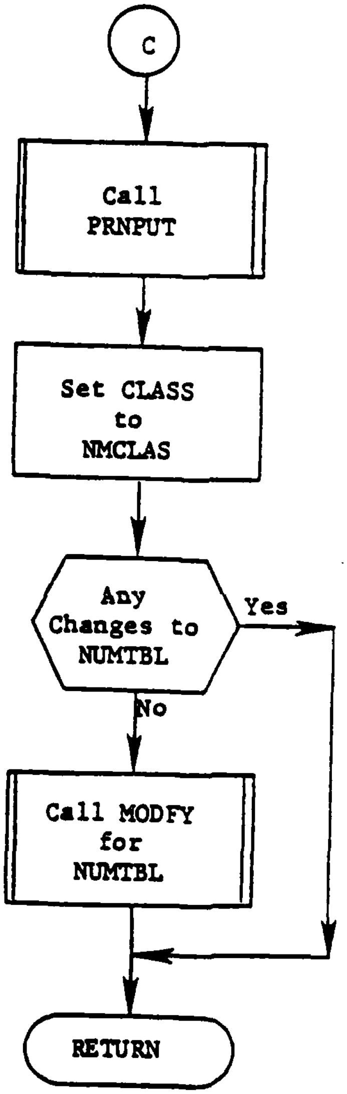


Figure 3. (Part 4 of 4)

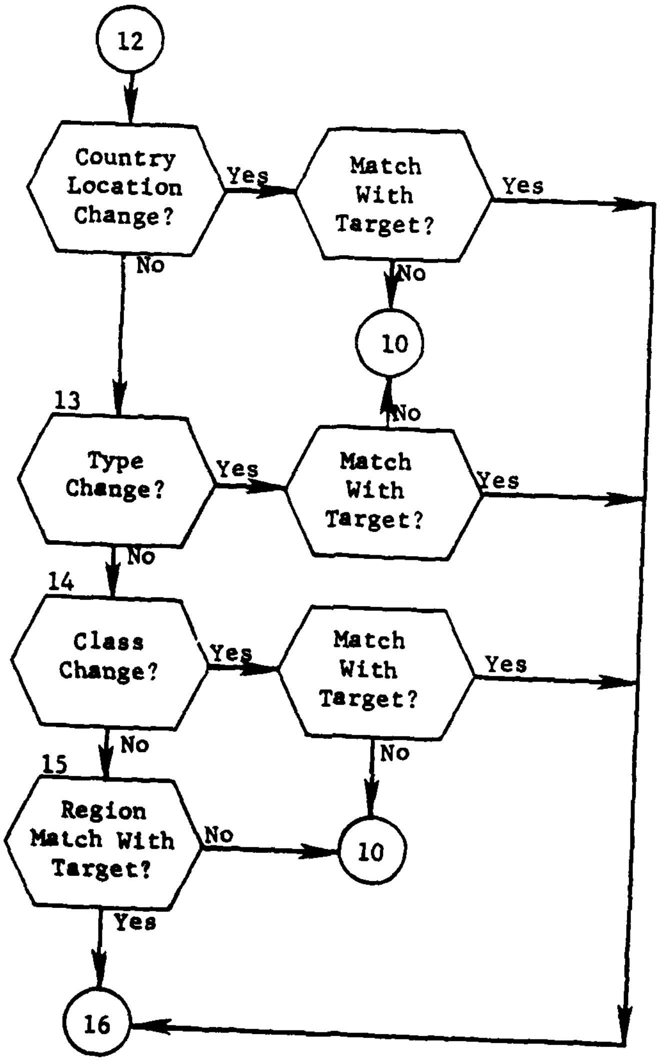


Figure 4. (Part 6 of 9)

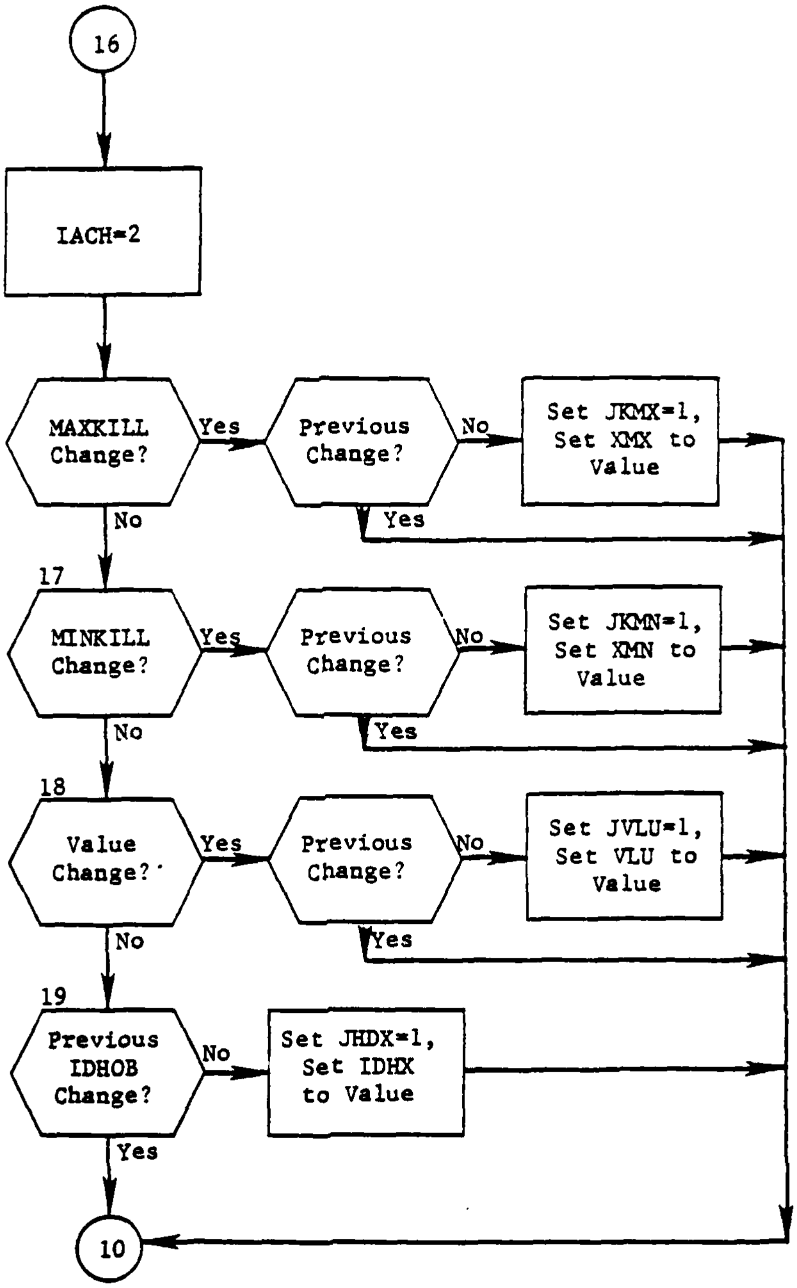


Figure 4. (Part 7 of 9)

2.9 Subroutine FIXWEP

PURPOSE:	To create fixed weapon assignments
ENTRY POINTS:	FIXWEP
FORMAL PARAMETERS:	ICPT - index of FIX clause
COMMON BLOCKS:	C10, C15, C30, FIXOFF, IFXREQ, OOPS, TSTUFF, WEPCOM, ZEES
SUBROUTINES CALLED:	DIRECT, DISTF, HDFND, HEAD, INSGET, MODFY, NEXTTT, RETRV, STORE, TOFM, WEPIN
CALLED BY:	ENTMOD (of PREPALOC)

Method:

Thc first step in the FIXWEP process is to retrieve the target list header (TARNUM) and initialize break-point tables. These break-point tables are used to speed retrieval of records on the LISTXX chain. By saving the direct reference code of up to 500 records evenly spaced along the chain, any record may be retrieved by beginning the search at the nearest break-point record that precedes it.

The remainder of the process is driven by the FIX clause which caused the call. The FIX clause is viewed as a series of subclauses each of which contains a subject and one or more objects. The subject is a collection of attributes and each object is a collection of values for those attributes. As a subject is read, its component attributes are used to set up a group of switches to indicate the operations to be carried out for each object. The switches are:

ISRNG - 1 if one DESIG, 2 if two DESIGs
IASET - 1 if no set arrival time, 2 if arrival time is set

FIXWEP then reads in each related object and creates ASSIGN records as follows. Each object will set the following values:

IXGRP - the group number
ISTSAL - the selected salvo (0 if non selected)
AXRV - the selected arrival time (0 if IASET = 1)
INMA - the number of assignments (set by NUMALOC)
CDSG - the DESIG of the target

If there is a DESIG range (ISRNG = 2) the values of the range are placed in CDSG in order and the process carried out for each.

For each set of the above values, FIXWEP first checks to see if the current group (IGRP) is the same as IXGRP. If not, the current group is modified and the new group retrieved. The target whose DESIG is CDSG is not obtained and the corresponding TARCDE record found through use of the break-point tables. Now the process loops INMA times. In each loop FIXWEP first checks to see if group resources are exceeded. If not, the process proceeds depending upon whether or not the group is salvoed. For a non-salvoed group FIXWEP simply stores a new ASSIGN record.

For a salvoed group the salvo is determined in one of three ways. If neither a salvo nor an arrival time has been specified, a convenient salvo is selected. If a salvo was specified, it is used. If an arrival time was specified, the time of flight is calculated and the salvo set to fulfill the desired time of arrival. In any case, FIXWEP now checks to see if a weapon is available in the chosen salvo. If so, the ASSIGN record is stored. If not, FIXWEP finds a salvo that is available and uses that salvo instead unless the desired salvo was specified. In the later case, FIXWEP will search all current assignments for the weapon group with the desired salvo and shift to the available salvo that was either assigned with no specification or, failing to find such, one whose arrival time was specified. The new assignment may then be made to the desired salvo.

Since IREG is an attribute on both the WEPNGP and the TARGET records it must be saved in IRGHOLD after CALL DIRECT ('WEPNGP') then restored before CALL MODFY ('WEPNGP'). Otherwise, IREG may contain a value from the TARGET and degrade the IDS structure for the WEPNGP record, since IREG is a control key for it.

Subroutine FIXWEP is illustrated in figure 5.

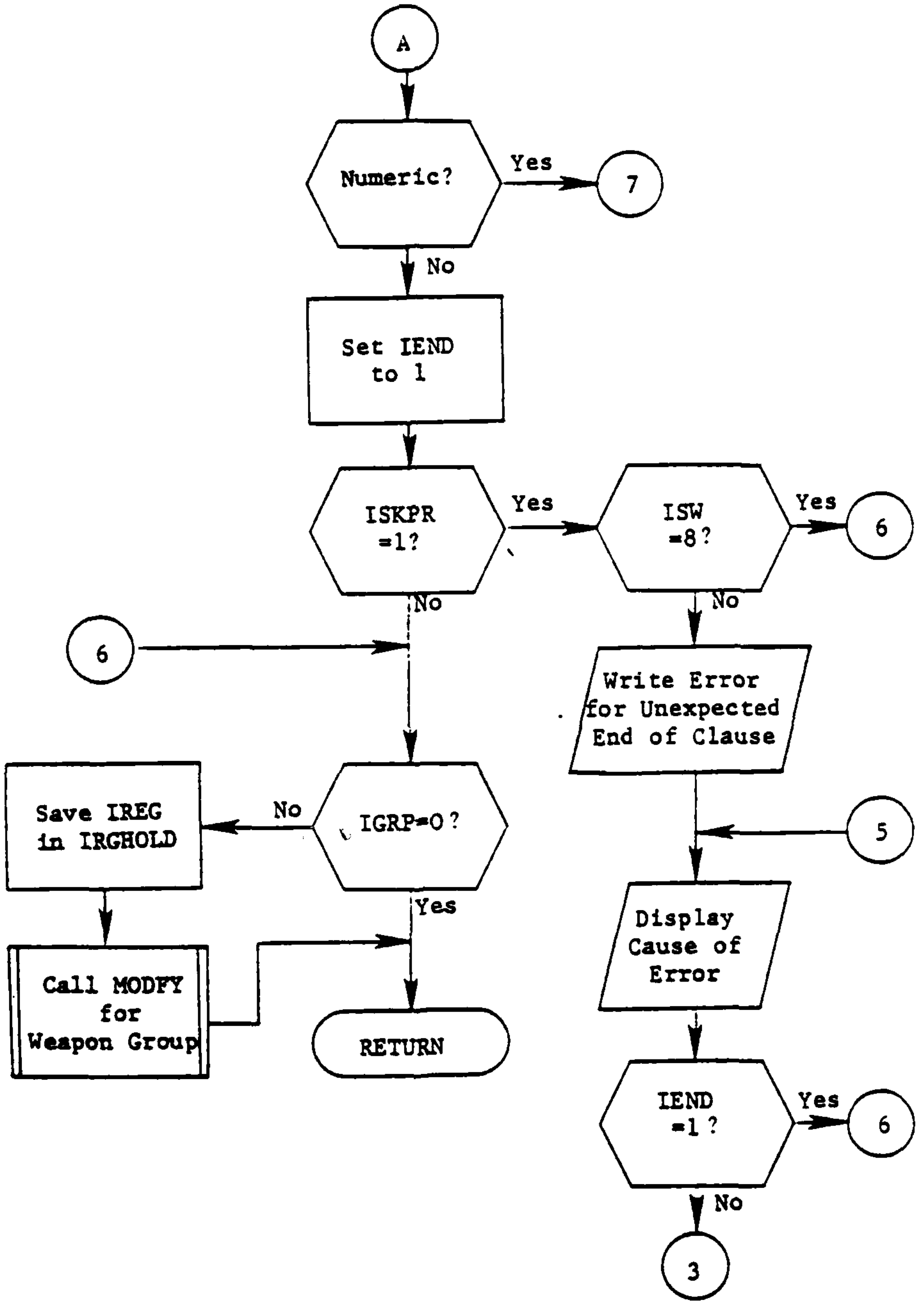


Figure 5. (Part 3 of 23)

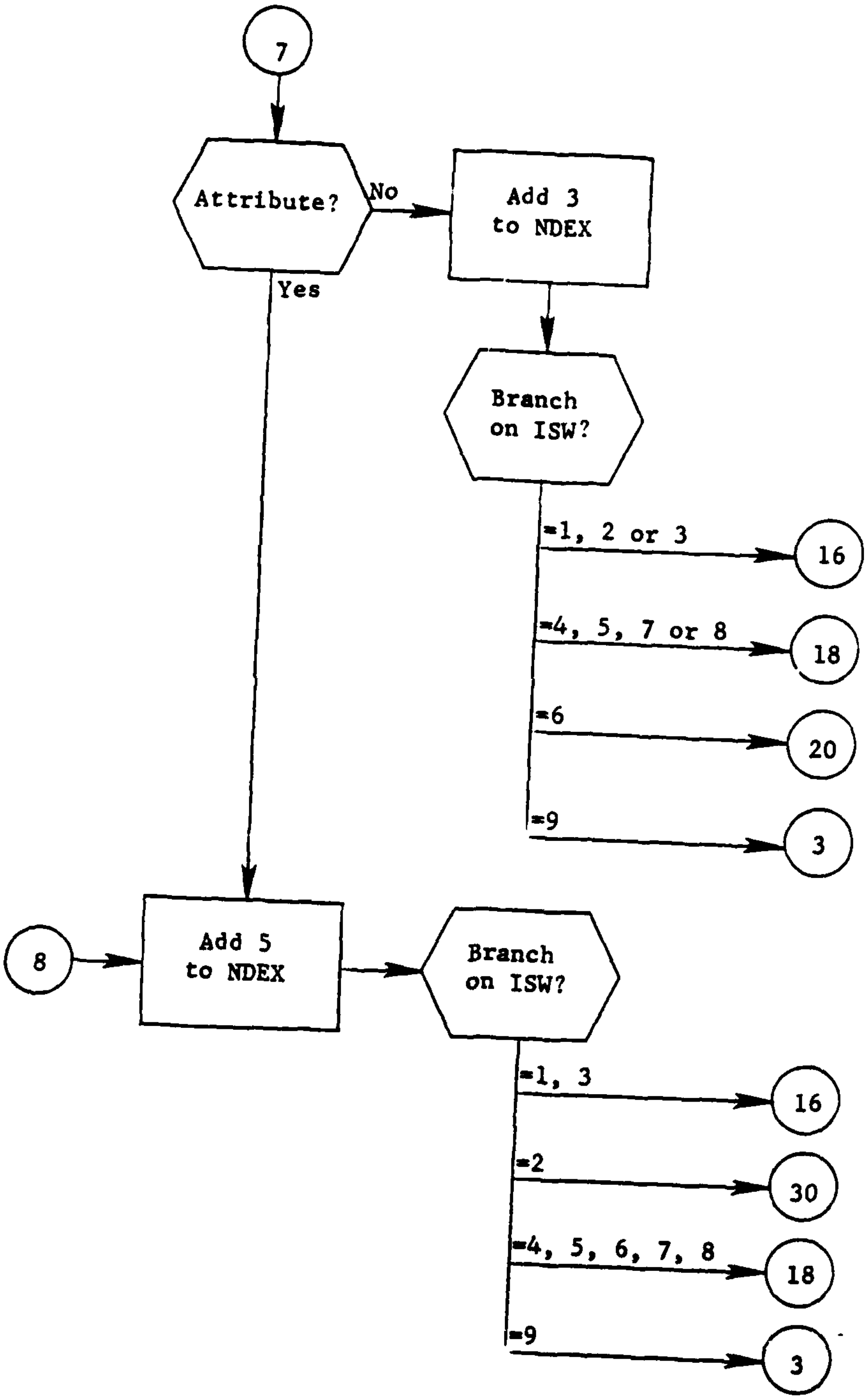


Figure 5. (Part 4 of 23)

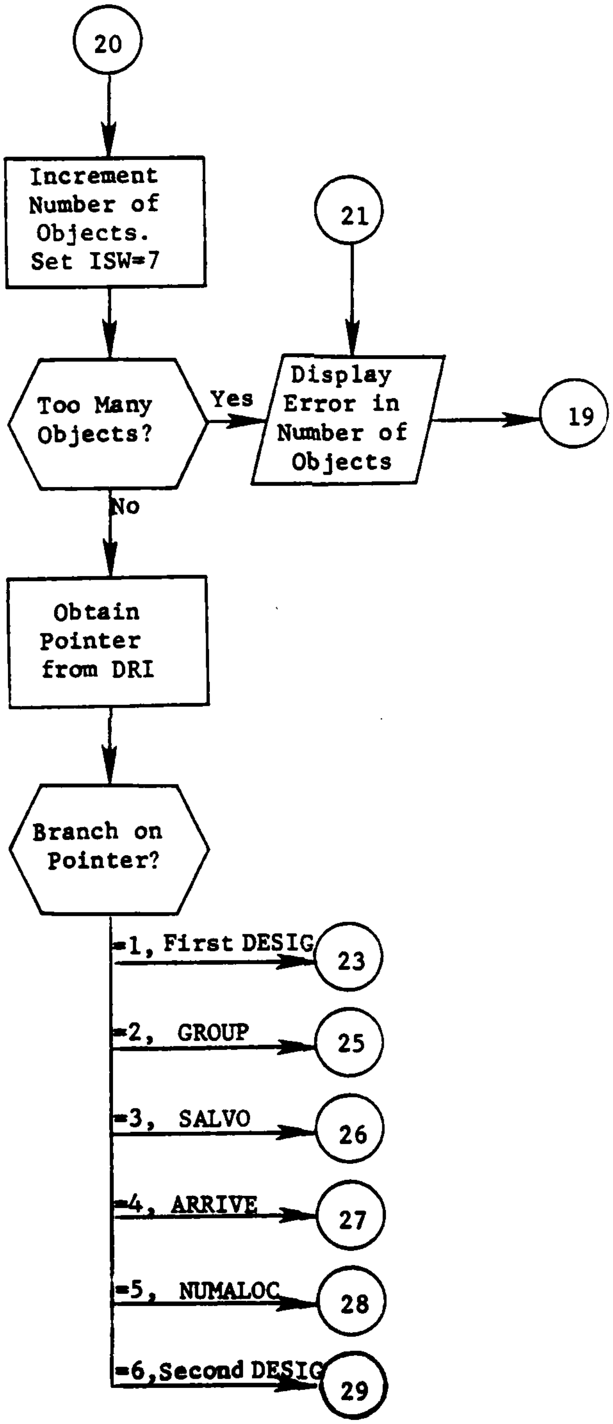


Figure 5. (Part 11 of 23)

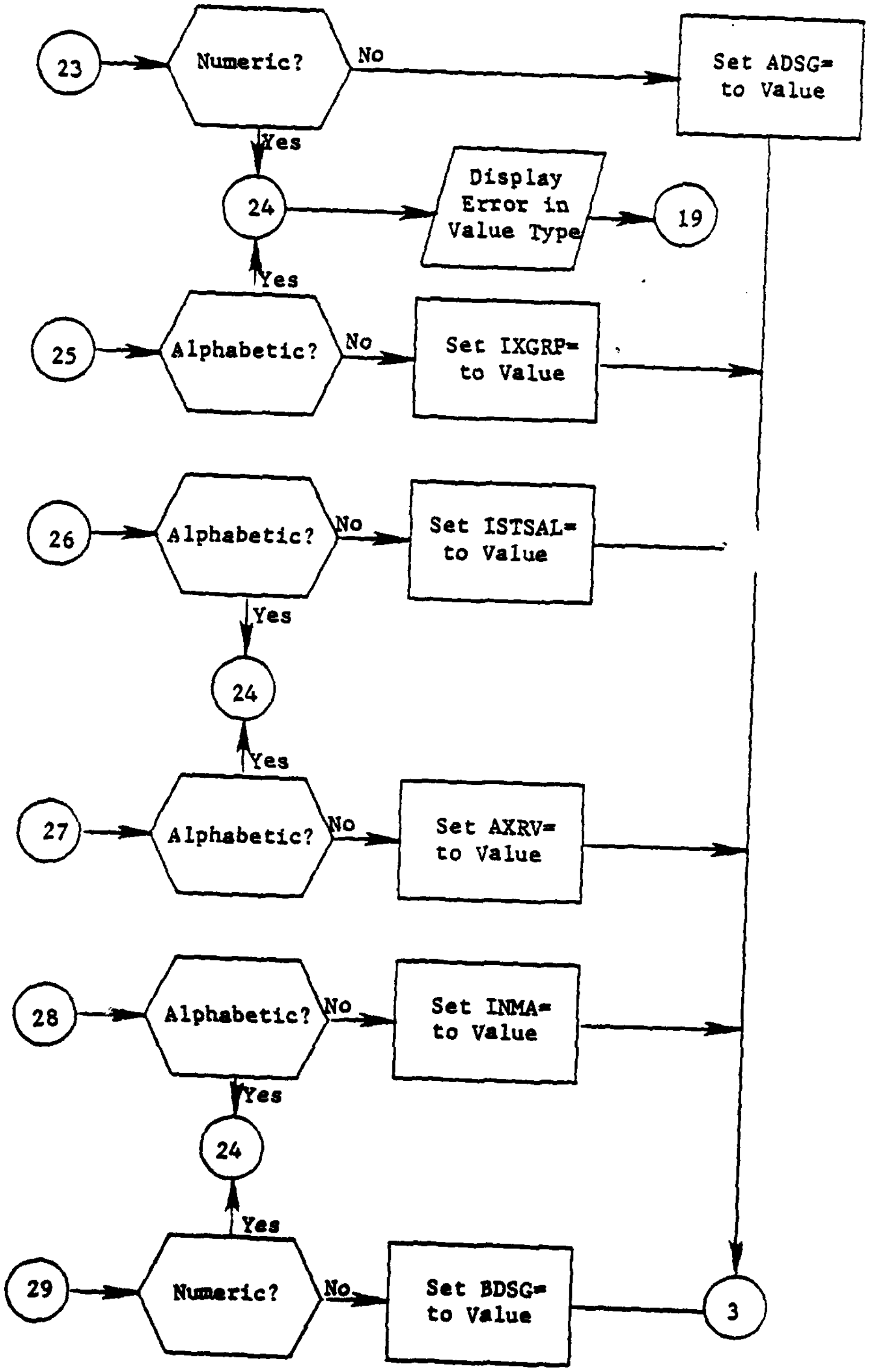


Figure 5. (Part 12 of 23)

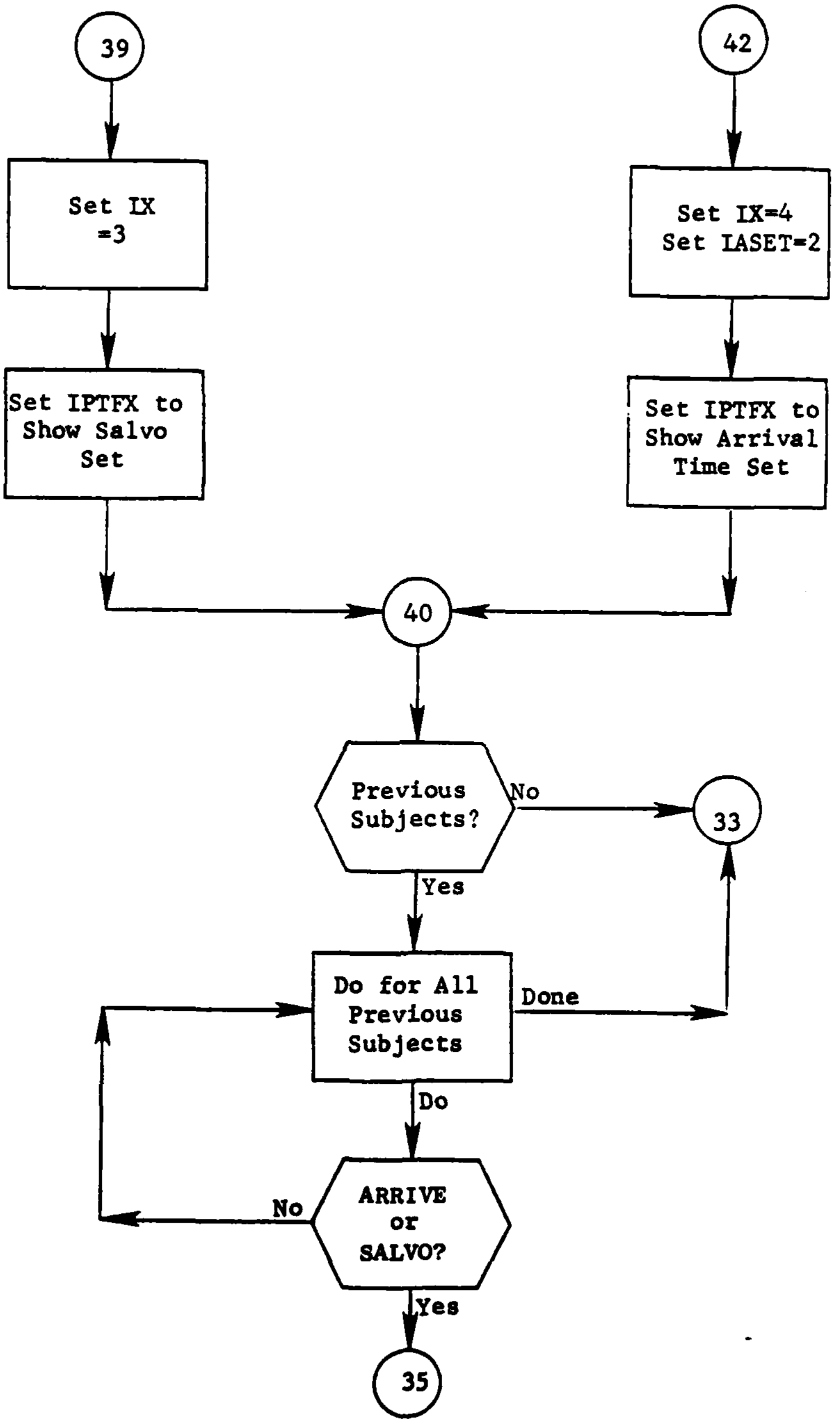

39
Set IX =3
Set IPTFX to Show Salvo Set
42
Set IX=4 Set IASET=2
Set IPTFX to Show Arrival Time Set
40
Previous Subjects?
No
33
Yes
Do for All Previous Subjects
Done
Do
ARRIVE or SALVO?
No
Yes
35

Figure 5. (Part 15 of 23)

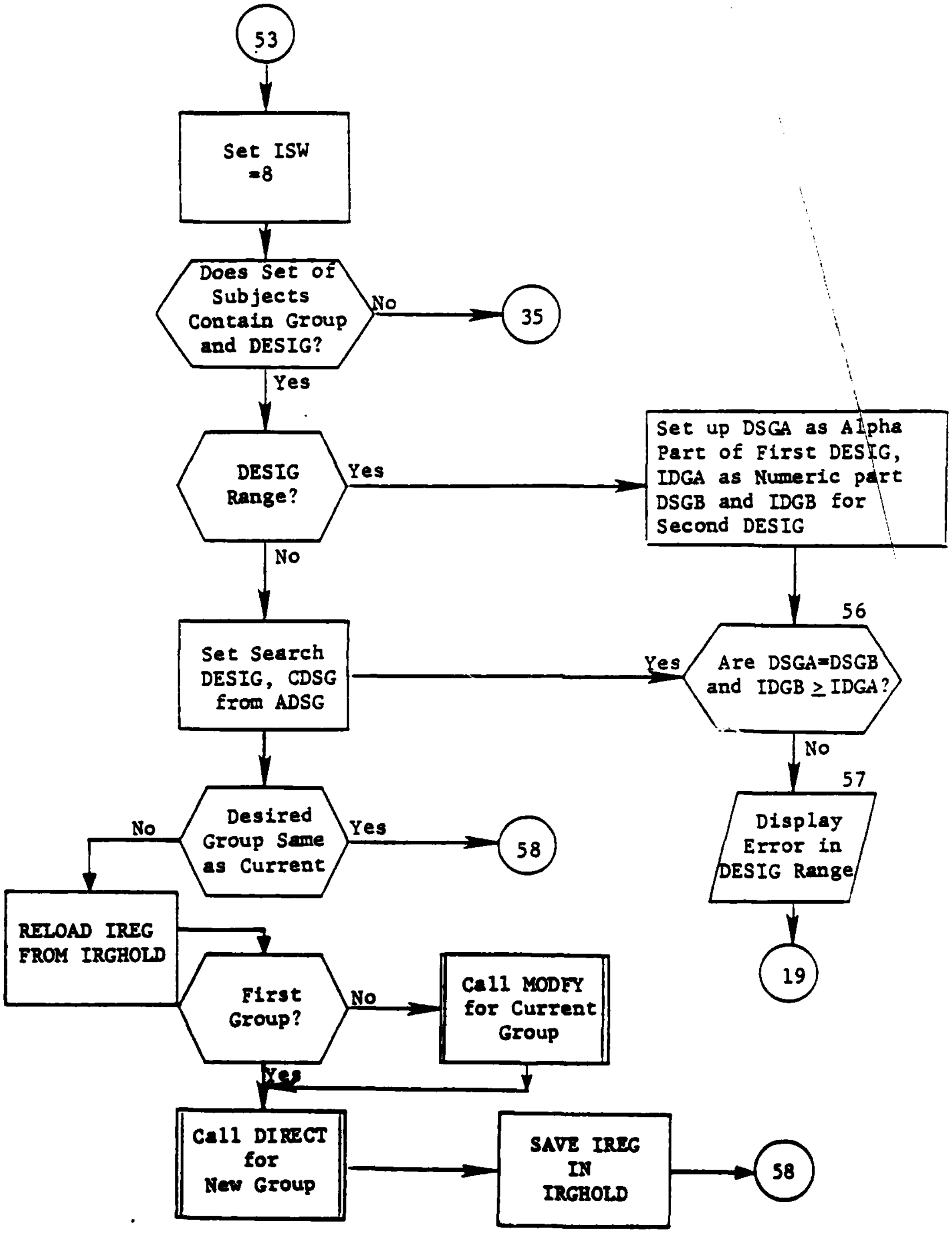

53
Set ISW =8
Does Set of Subjects Contain Group and DESIG?
No
35
Yes
DESIG Range?
Yes
Set up DSGA as Alpha Part of First DESIG, IDGA as Numeric part DSGB and IDGB for Second DESIG
No
Set Search DESIG, CDSG from ADSG
56
Yes
Are DSGA=DSGB and IDGB ≥ IDGA?
No
57
Display Error in DESIG Range
19
Desired Group Same as Current
No
Yes
58
RELOAD IREG FROM IRGHOLD
First Group?
No
Call MODFY for Current Group
Yes
Call DIRECT for New Group
SAVE IREG IN IRGHOLD
58

Figure 5. (Part 16 of 23)

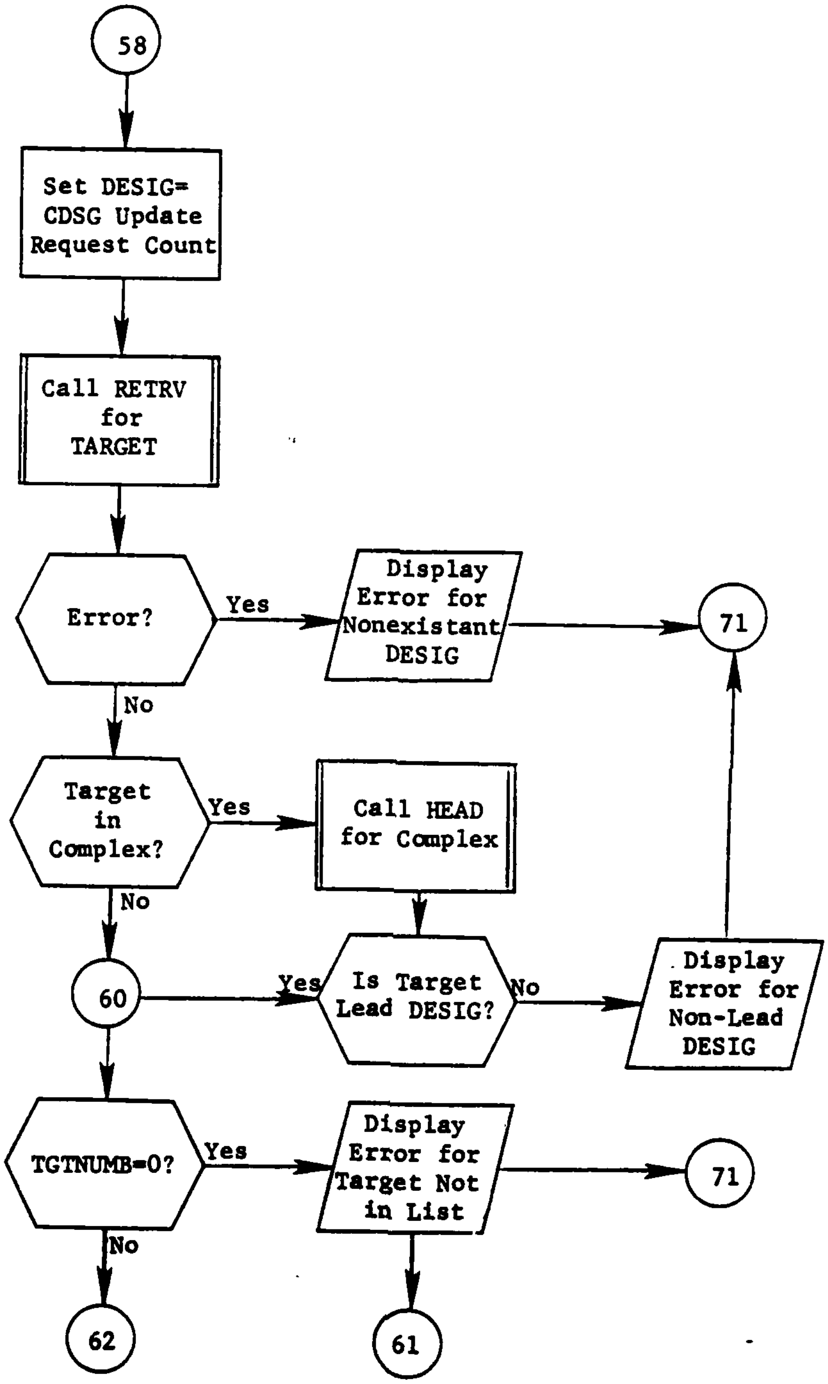

58
Set DESIG= CDSG Update Request Count
Call RETRV for TARGET
Error?
Yes
Display Error for Nonexistant DESIG
71
No
Target in Complex?
Yes
Call HEAD for Complex
No
60
Yes
Is Target Lead DESIG?
No
Display Error for Non-Lead DESIG
TGTNUMB=0?
Yes
Display Error for Target Not in List
71
No
62
61

Figure 5. (Part 17 of 23)

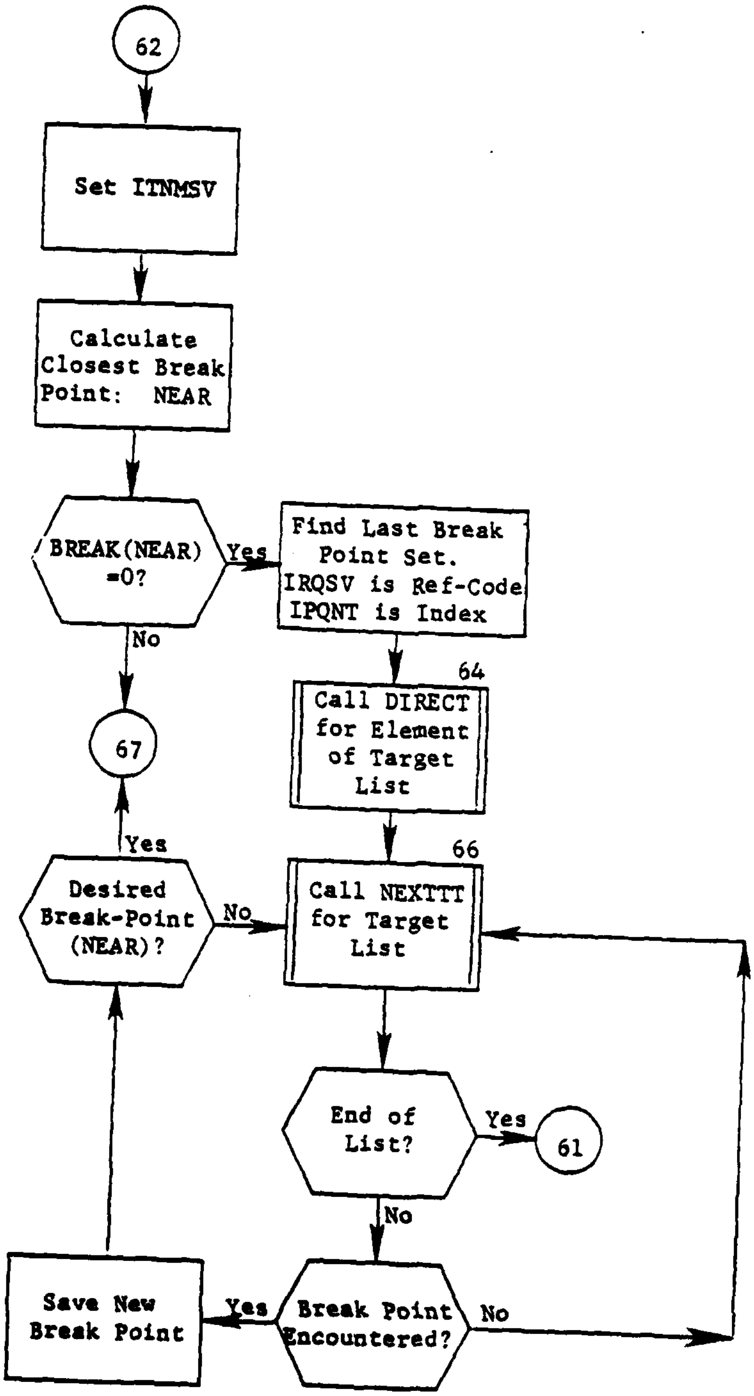

62
Set ITNMSV
Calculate Closest Break Point: NEAR
BREAK(NEAR) =0?
Yes
Find Last Break Point Set. IRQSV is Ref-Code IPQNT is Index
No
64
Call DIRECT for Element of Target List
67
Yes
66
Desired Break-Point (NEAR)?
No
Call NEXTTT for Target List
End of List?
Yes
61
No
Save New Break Point
Yes
Break Point Encountered?
No

Figure 5. (Part 18 of 23)

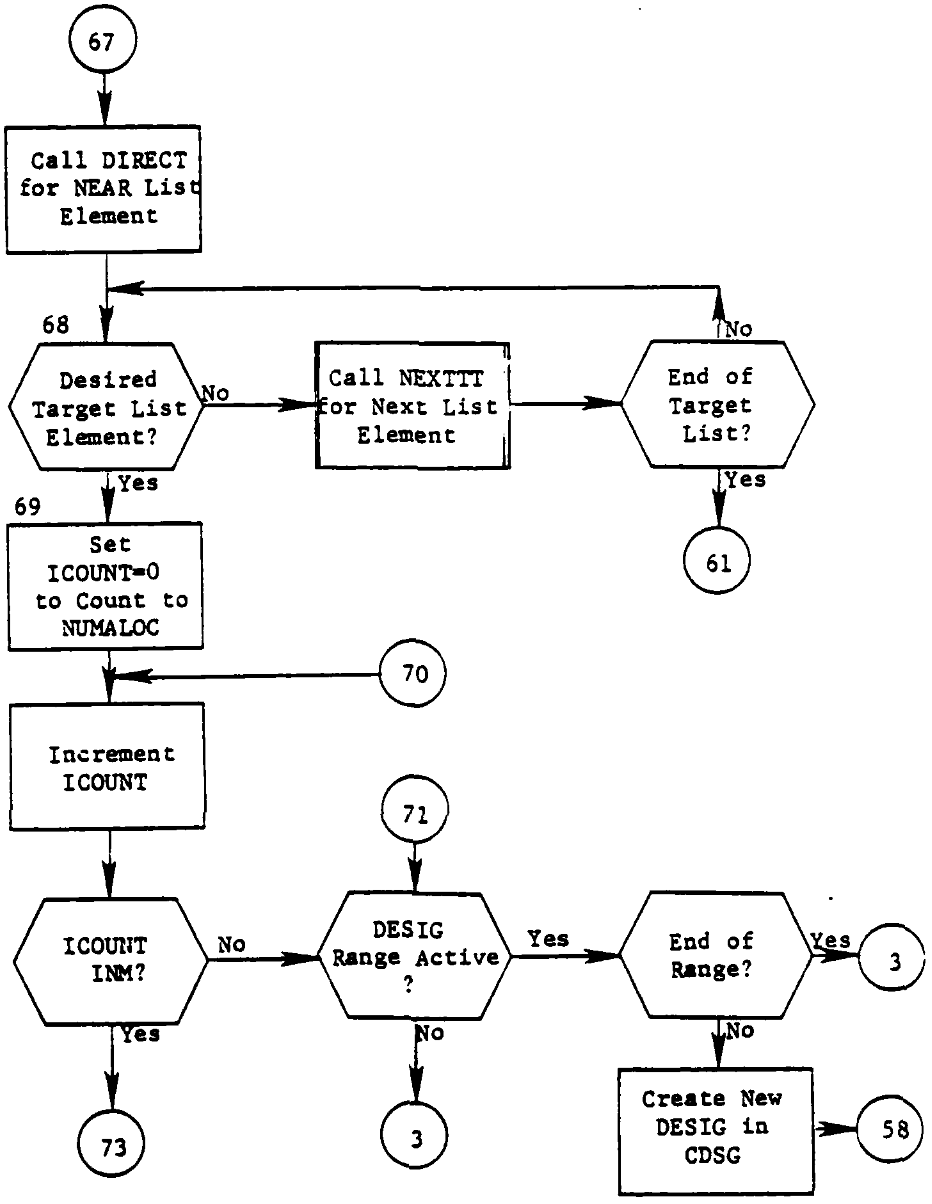


Figure 5. (Part 19 of 23)

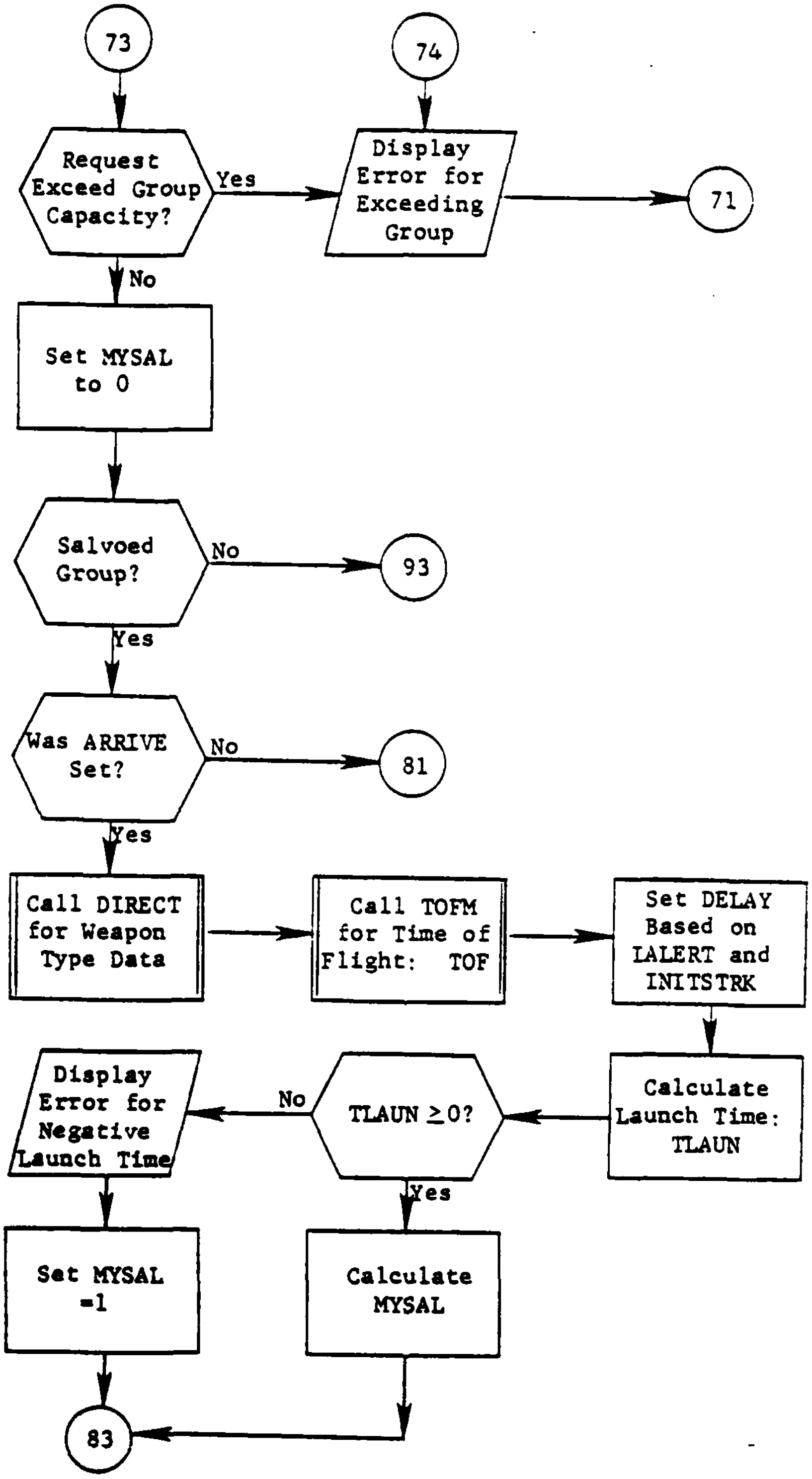

73
74
Request Exceed Group Capacity?
Yes
Display Error for Exceeding Group
71
No
Set MYSAL to 0
Salvoed Group?
No
93
Yes
Was ARRIVE Set?
No
81
Yes
Call DIRECT for Weapon Type Data
Call TOFM for Time of Flight: TOF
Set DELAY Based on IALERT and INITSTRK
Calculate Launch Time: TLAUN
TLAUN ≥ 0?
No
Display Error for Negative Launch Time
Yes
Set MYSAL =1
Calculate MYSAL
83

Figure 5. (Part 20 of 23)

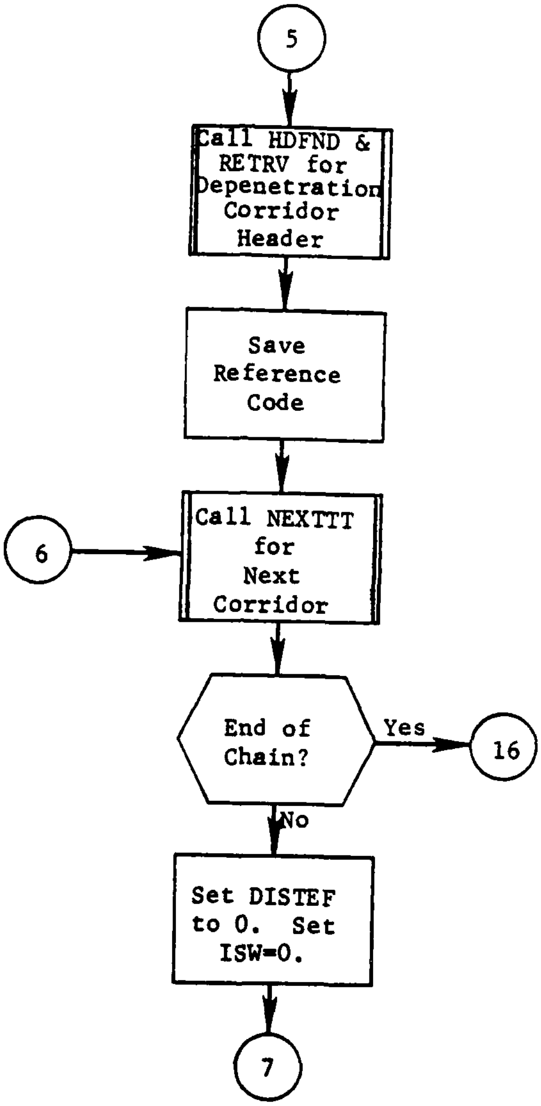

5
Call HDFND & RETRV for Depenetration Corridor Header
Save Reference Code
6
Call NEXTTT for Next Corridor
End of Chain?
Yes
16
No
Set DISTEF to 0. Set ISW=0.
7

Figure 6. (Part 3 of 7)

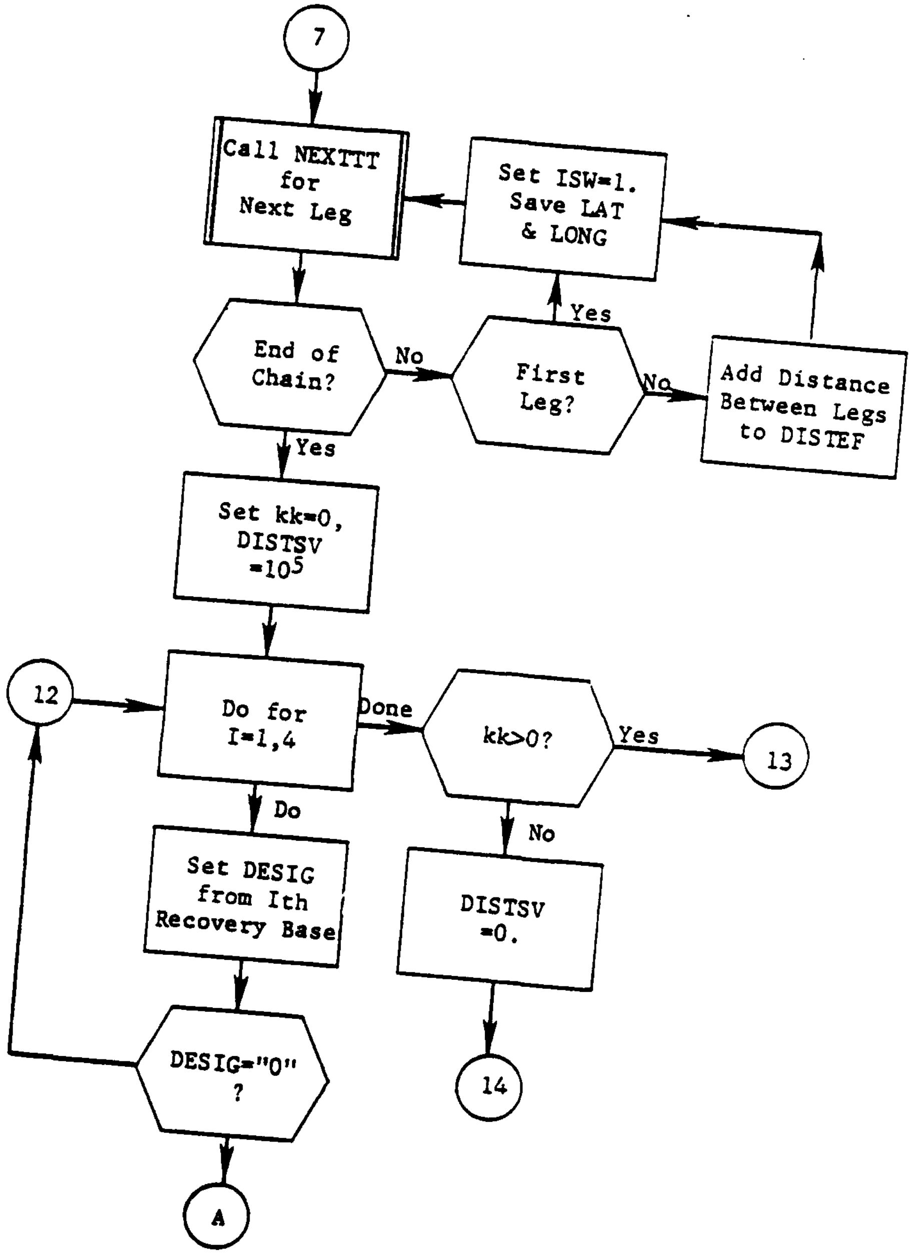


Figure 6. (Part 4 of 7)

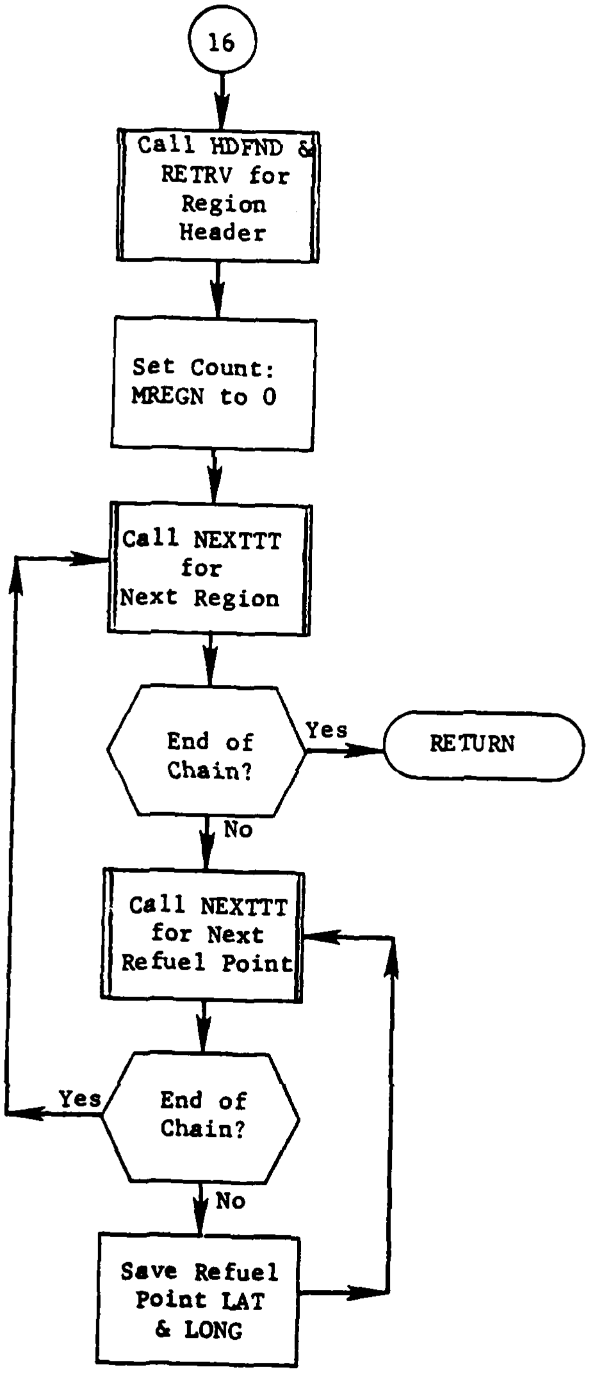

16
Call HDFND & RETRV for Region Header
Set Count: MREGN to 0
Call NEXTTT for Next Region
End of Chain?
Yes
RETURN
No
Call NEXTTT for Next Refuel Point
End of Chain?
Yes
No
Save Refuel Point LAT & LONG

Figure 6. (Part 7 of 7)

2.11 Subroutine GEOPREP

PURPOSE: To calculate and store geographic data

ENTRY POINTS: GEOPREP

FORMAL PARAMETERS: None

COMMON BLOCKS: C10, C15, C30, CRLENGTH, DISTEF, GEOREF, OOPS

SUBROUTINES CALLED: DIRECT, DISTF, DLETE, GEOIN, HDFND, HEAD, MODFY, NEXTTT, RETRV, STORE

CALLED BY: ENTMOD (of PREPALOC)

Method:

Depenetration corridors are processed first. All old TDDIST and RDDIST records are deleted. Then a new RDDIST record is stored connecting the corridors with their recovery bases. Next, penetration corridors are processed. All old TPDIST records are deleted. Then all attributes sectors are calculated and the penetration corridor record (PENCRD) modified.

The subroutine now cycles through the target list. For each target it finds the shortest depenetration distance, taking into account the distance to the corridor, the length of the corridor and the distance from the corridor to the recovery base. A TDDIST record is created for the best depenetration corridor. A TPDIST record is also stored connecting the target list element to every penetration corridor.

Subroutine GEOPREP is illustrated in figure 7.

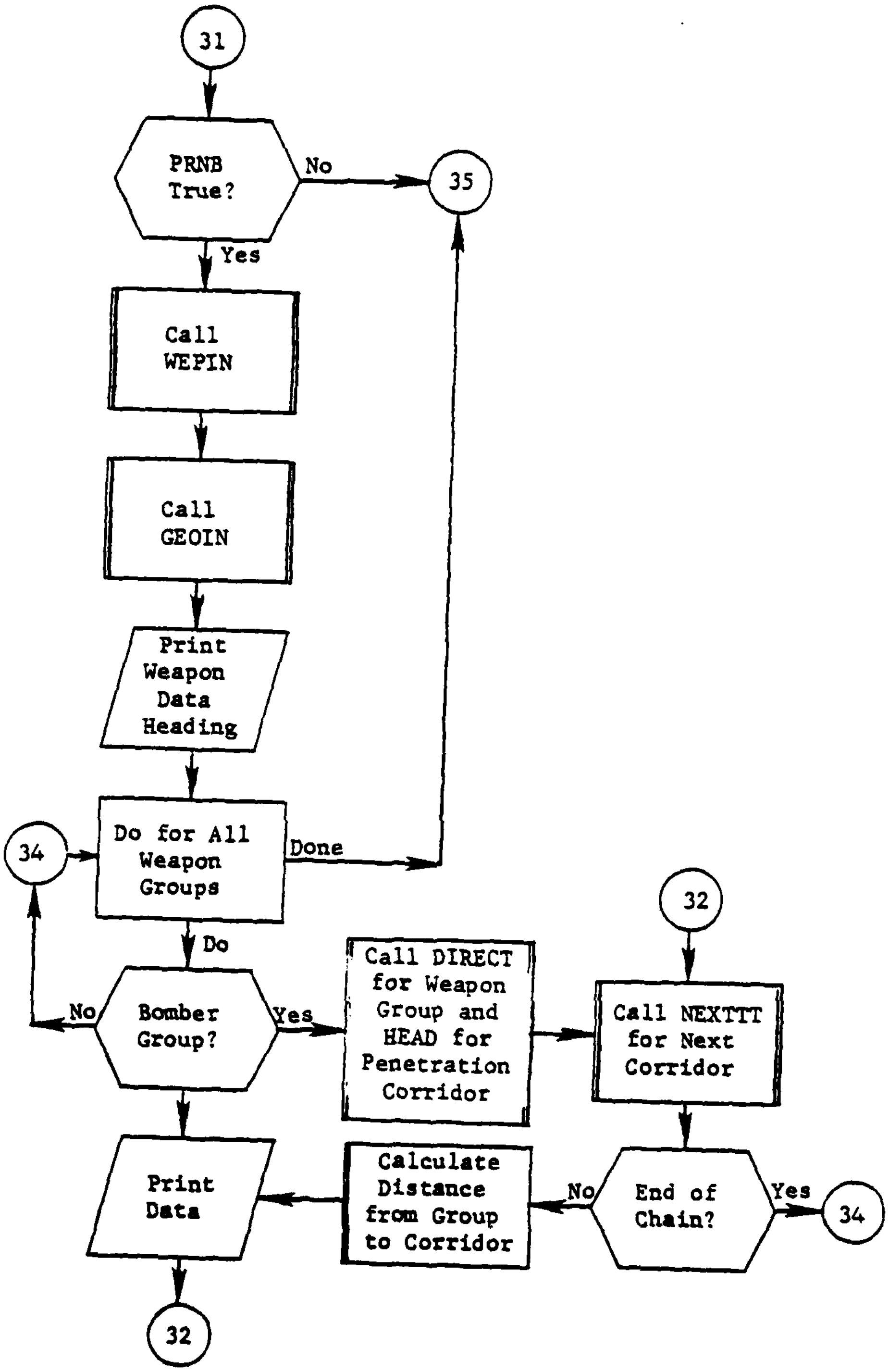

31
PRNB True?
No
35
Yes
Call WEPIN
Call GEOIN
Print Weapon Data Heading
34
Do for All Weapon Groups
Done
32
Do
No
Bomber Group?
Yes
Call DIRECT for Weapon Group and HEAD for Penetration Corridor
Call NEXTTT for Next Corridor
Print Data
Calculate Distance from Group to Corridor
No
End of Chain?
Yes
34
32

Figure 8. (Part 8 of 12)

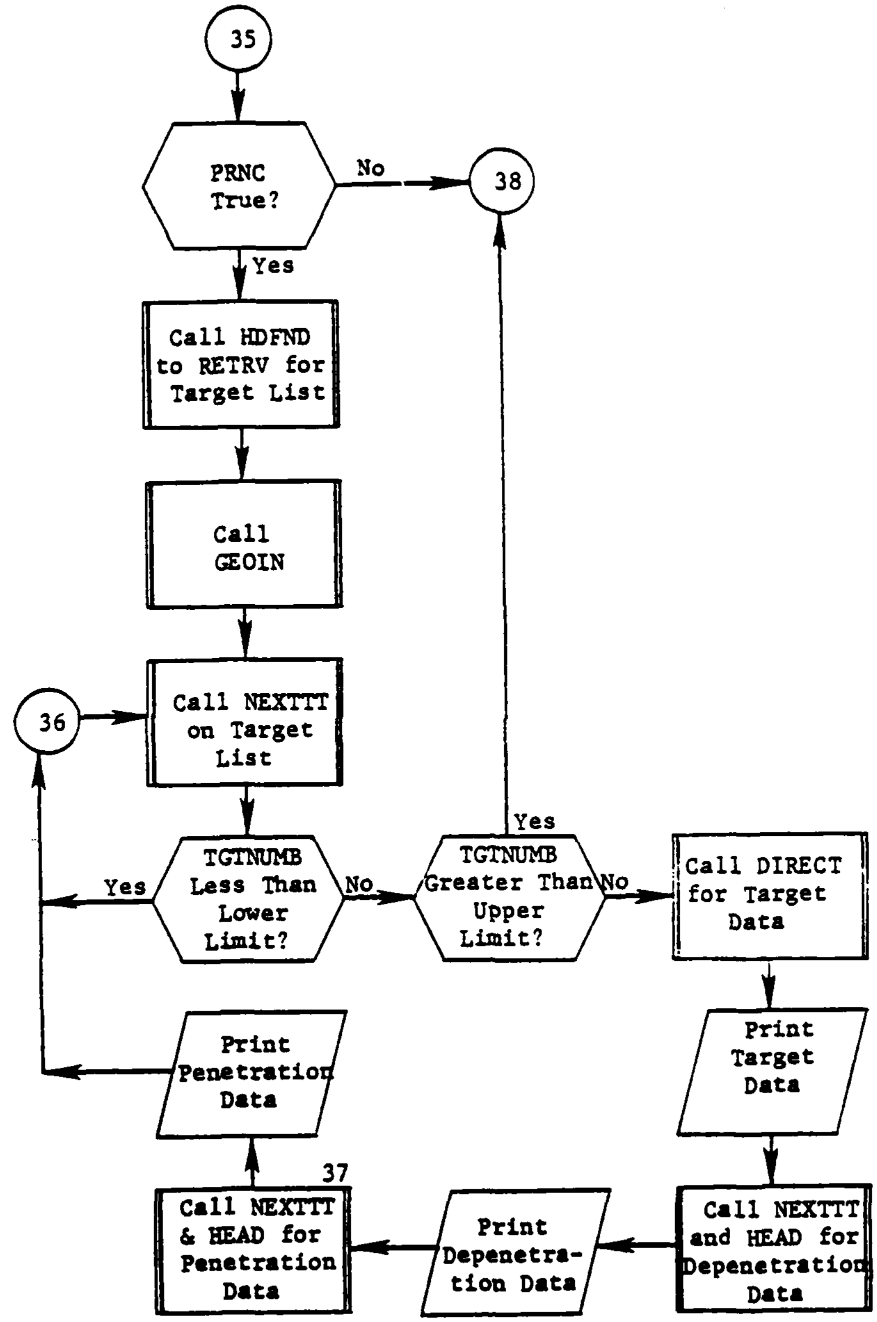

35
PRNC True?
No
38
Yes
Call HDFND to RETRV for Target List
Call GEOIN
36
Call NEXTTT on Target List
TGTNUMB Less Than Lower Limit?
Yes
No
TGTNUMB Greater Than Upper Limit?
Yes
No
Call DIRECT for Target Data
Print Target Data
Call NEXTTT and HEAD for Depenetration Data
Print Depenetration Data
37
Call NEXTTT & HEAD for Penetration Data
Print Penetration Data

Figure 8. (Part 9 of 12)

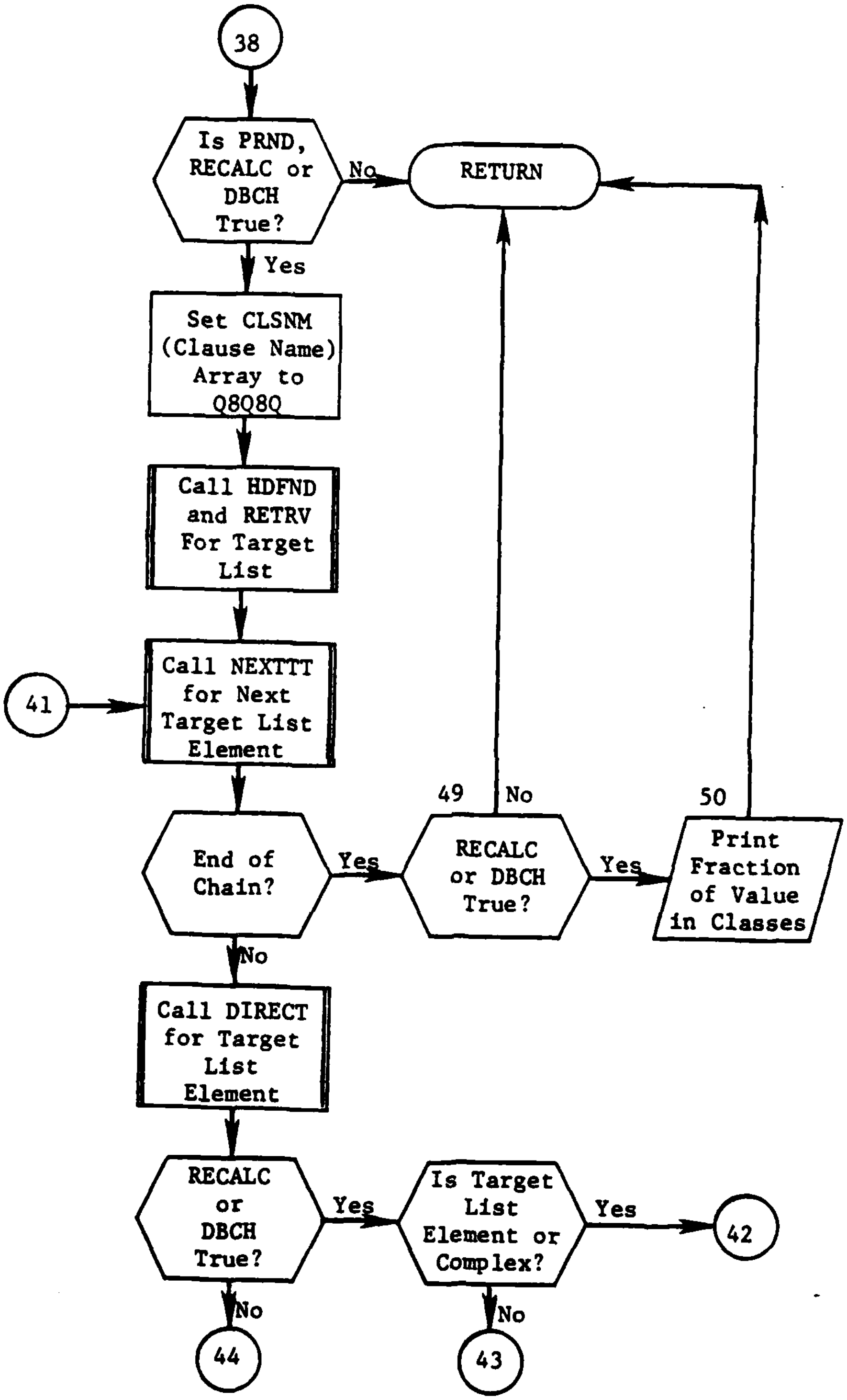

38
Is PRND, RECALC or DBCH True?
No
RETURN
Yes
Set CLSNM (Clause Name) Array to Q8Q8Q
Call HDFND and RETRV For Target List
41
Call NEXTTT for Next Target List Element
End of Chain?
Yes
49
No
RECALC or DBCH True?
Yes
50
Print Fraction of Value in Classes
No
Call DIRECT for Target List Element
RECALC or DBCH True?
Yes
Is Target List Element or Complex?
Yes
42
No
44
No
43

Figure 8. (Part 10 of 12)

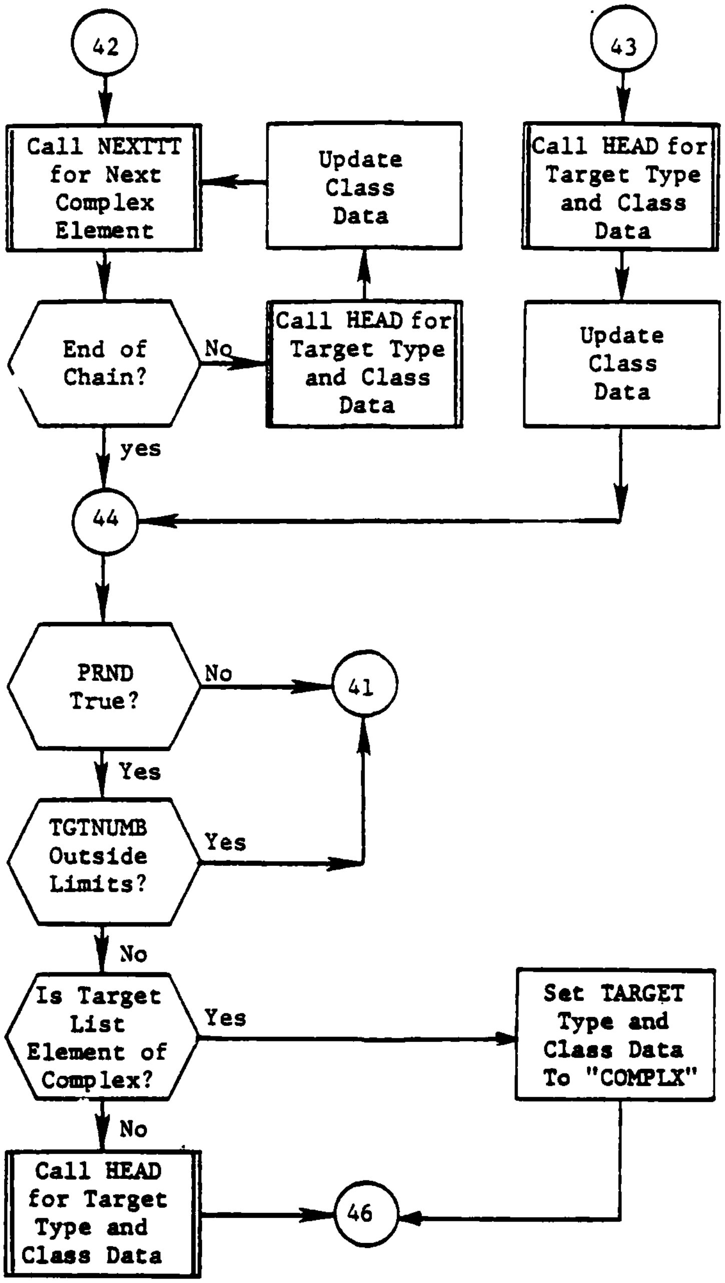

42
43
Call NEXTTT for Next Complex Element
Update Class Data
Call HEAD for Target Type and Class Data
End of Chain?
No
Call HEAD for Target Type and Class Data
Update Class Data
yes
44
PRND True?
No
41
Yes
TGTNUMB Outside Limits?
Yes
No
Is Target List Element of Complex?
Yes
Set TARGET Type and Class Data To "COMPLX"
No
Call HEAD for Target Type and Class Data
46

Figure 8. (Part 11 of 12)

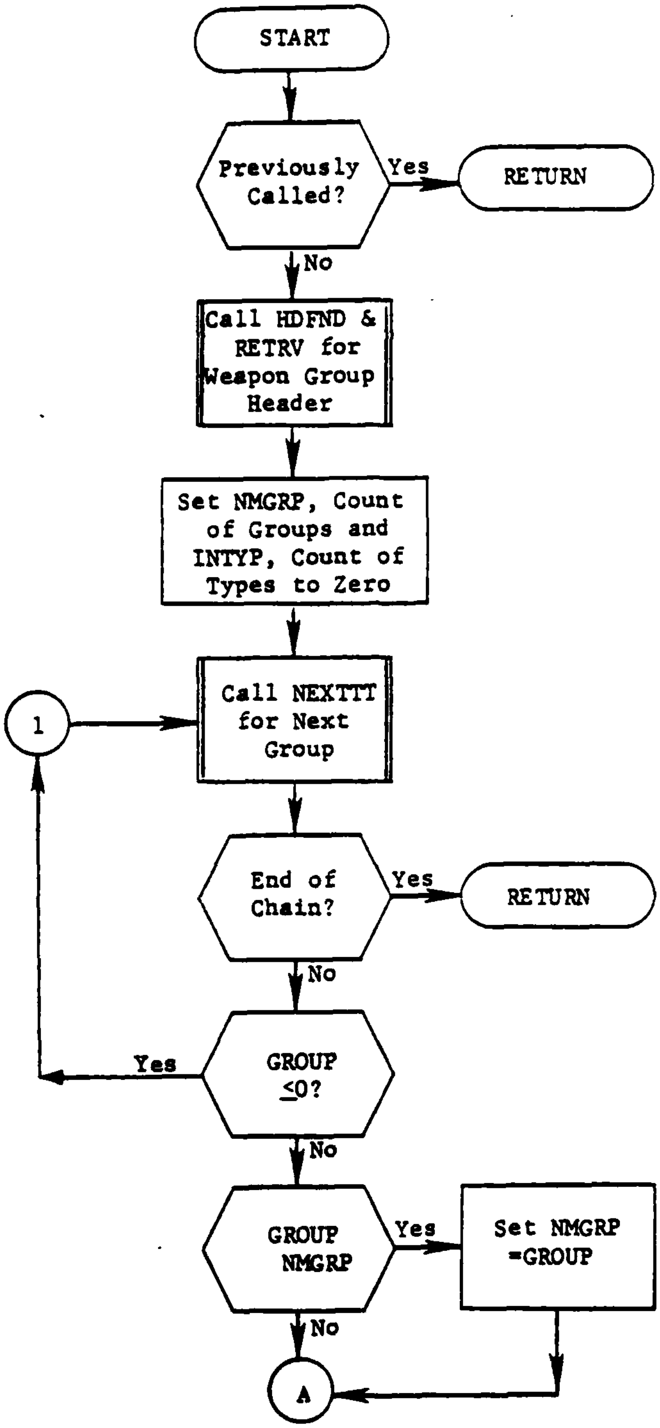

START
Previously Called?
Yes
RETURN
No
Call HDFND & RETRV for Weapon Group Header
Set NMGRP, Count of Groups and INTYP, Count of Types to Zero
1
Call NEXTTT for Next Group
End of Chain?
Yes
RETURN
No
Yes
GROUP ≤0?
No
GROUP NMGRP
Yes
Set NMGRP =GROUP
No
A

Figure 10. Subroutine WEPIN (Part 1 of 3)

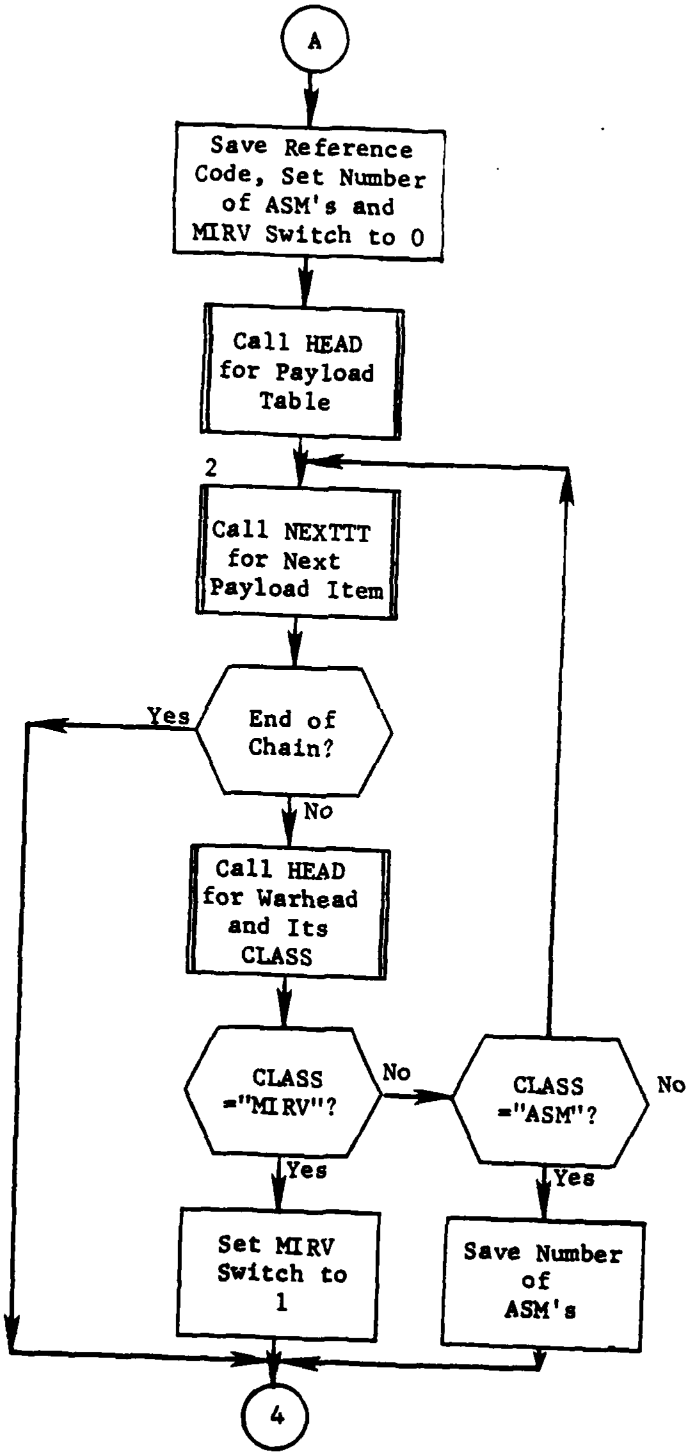

A
Save Reference Code, Set Number of ASM's and MIRV Switch to 0
Call HEAD for Payload Table
2
Call NEXTTT for Next Payload Item
End of Chain?
Yes
No
Call HEAD for Warhead and Its CLASS
CLASS ="MIRV"?
No
Yes
CLASS ="ASM"?
No
Yes
Set MIRV Switch to 1
Save Number of ASM's
4

Figure 10. (Part 2 of 3)

SECTION 3. ALOC MODULE

3.1 Purpose

The major purpose of this module is to determine the optimal allocation of weapons to targets, using a Lagrange multiplier technique. The weapons are divided into weapon groups -- each group containing weapons of the same characteristics which are geographically close. Thus, except for time launch interval constraints for some "salvoed" missiles, weapons are considered identical within groups. Each target is considered individually for weapon assignment. The order of investigation is the order of the TARCDE records on the LISTXX chain which was determined by the PLANSET module. When all targets have been processed, another pass over this chain begins. This process continues until the Lagrange method has allocated all the weapons to targets. The assignments are stored on a temporary file during the process and entered into the integrated data base or ASSIGN records when allocation is complete.

The user is able to specify weapon assignments through the FIX adverb to the PREPALOC module. In this case the ALOC module will optimally assign those weapons which have not been fixed. In addition, there are capabilities which allow the user to modify weapon range values, to restrict the use of MIRV weapons by target class, and to restrict the use of any weapon group by the value of either of the target attributes *FLAG* or *CNTRYL*.

3.2 Input

The precondition of the integrated data base required is that the PREPALOC module has to have been executed. Furthermore, there is an optional input file -- the Weapon/Target Data File. The Weapon/Target Data file contains the information relating each weapon group to each target. The Weapon/Target Data file, if not input, is created by the FRSTGD subroutine on pass one and may be retained for later executions of ALOC. Two record types are alternated on the file. The first record of each pair contains target data. The record length is fixed and its contents are shown in table 2. The second record contains information concerning the target whose data are on the previous record and each of the weapon groups in the data base. The following example will be used to describe the contents and format of this record. Assume the target on the first record had two hardness components (HC1 and HC2) and is defended by terminal missiles defenses. There are five attacking groups as shown in figure 10.2.

The record contains four basic data items. The first set of data is the bomber penetration corridor for each bomber group. This data is packed using five bits for each entry as shown in table 2.1. The second set of

Table 2. Format of Target Data Record - File Code 15

Word	Content
1	Target Class
2	Target Type
3-7	Time sensitive target input times
8-12	Time sensitive target values
13	Country location
14	Country owner
15	Flag
16	Number of hardness components
17	Number of time components
18	Region
19	Complex indicator (ICOMPL)
20	Latitude
21	Longitude
22	First hardness component radius for ground burst
23	Second hardness component radius for ground burst
24	First hardness component radius for air burst
25	Second hardness component radius for ground burst
26	MAXKILL
27	MINKILL
28	Maximum value of weapons to be used on target
29	Index number
30	Target radius
31	SIOP table number
32	Value of target
33	Normalized target value
34	Height of burst indicator
35	High altitude target defense
36	Low altitude target defense
37	Missile defense
38	Number of terminal interceptors
39	Target name
40	DESIG
41	Value of first hardness component
42	Value of second hardness component

Group	B1	M1	M2	B2	B3
Type	Bomber	Missile	Missile	Bomber	Bomber
Payload	Bombs	MIRV	MRV	Bomb & ASM	ASM
Penetration Probability	PB1	-	-	PB2	PB3
First Weapon	Bomb	RV	AY*	Bomb	ASM
Second Weapon	-	-	RV	ASM	-

*Note: AY - Composite yield of MRV system is used.

Figure 10.2. Target-Group Data

Table 2.1. Format of Weapon/Target Data Record - File Code 15

Bits Word	1 5 10 15 18	19 36
1	COR B1 COR B2 COB B3 ↔	Not Used↔
2	00↔Not Used↔	Not Used↔
3	TOA B1↔	TOA B2 ↔
4	TOA B3↔	TOA M1↔
5	TOA M2↔	PENPROB PB1↔
6	PENPROB PB2↔	PENPROB PB3↔
7(1)	PKB1 (First Weapon, HC1)↔	PKB1 (First Weapon, HC2)(2)
8	PKB2 (First Weapon, HC1)↔	PKB2 (First Weapon, HC2)↔
9	PKB3 (First Weapon, HC1)↔	PKB3 (First Weapon, HC2)↔
10	PKM1 (First Weapon, HC1)↔	PKM1 (First Weapon, HC2)↔
11	PKM2 (First Weapon, HC1)↔	PKM2 (First Weapon, HC2)↔
12	PKB2 (Second Weapon, HC1)(3)↔	PKB2 (First Weapon, HC2)↔
13	PKM2 (Second Weapon, HC1)(4)↔	PKM2 (Second Weapon, HC2)(4)

(1) If there are area missile defenses the penetration probabilities of the missiles would be entered at this point.

(2) If only one hardness component exists this field will be omitted and next data element will be placed in the field.

(3) If there is no second weapon this field would be filled by the next data element.

(4) If there is no missile defense these fields would be left blank.

data contains missile switches. If the missile is inactive, the switch is set to 1. The missile data always starts on a word boundary. The third set of data contains the time of arrival for both bomber and missile groups in that order. The fourth set of data consists of probabilities of kill. The sequence of the data is the probability of kill of the first weapon against the first target hardness component PK(1,HC1) and the probability of kill of the first weapon against the second target hardness component, PK(1,HC2). These pairs of data are stored for each group. This information is followed by the probability of kill of the second weapon, if it exists, against each hardness component, PK(2,HC1) and PK(2,HC2). If the target has no terminal defenses the second group of PKs is not stored for missiles. Data for the second hardness component is also omitted if only one hardness component of the target exists. In each of these two cases, the data fields are filled with the next data element. The record length depends on the number of bomber groups, number of missile groups, number of target hardness components and the existence of missile defenses. The lengths are calculated by the following expressions.

LENI(1) = LBC+LMS+(LUND+LS2U-1)/2+1	for one hardness component and no missile defense
LENI(2) = LBC+LMS+(LUND+2*LS2U-1)/2+1	for two hardness components and no missile defense
LENI(3) = LBC+LMS+(LDEF+LS2D-1)/2+1	for one hardness component and with missile defenses
LENI(4) = LBC+LMS+(LDEF+2*LS2D-1)/2+1	for two hardness components and with missile defenses

where:

LBC = (Number of bomber groups -1)/7+1
LMS = (Number of missile groups -1)/36+1
LUND = Number of groups + number of bomber groups (+ number of missile groups if there is a random area defense)
LDEF = 2 times the number of groups
LS2U = Number of groups + number of mixed loaded bomber groups
LS2D = LS2U + number of MRV missile groups

The Weapon/Target Data file is created in pass one and may also be used in subsequent runs (see RECALC Mode: Users Manual, UM 9-77, Volume III). The format for the Weapon/Target Data file -- file code 15 -- appears in tables 2 and 2.1.

3.3 Output

As a result of its execution the ALOC module creates ASSIGN (record type 70) records in the integrated data base. Further, the attribute NUMALOC is updated in every group to reflect the actual number of weapons allocated from that group.

3.4 Concept of Operation

In order to conserve storage, ALOC is broken up into two main overlays. The first overlay is called ALCINT. This overlay read; any user input, including restrictions, modifications, print request and so on. Furthermore, the weapon data is extracted from the integrated data base. The second overlay (ALCMUL) controls the determination of the allocation. The driver routine of this overlay is MULCON. Within the second overlay there are four segments. The first, FGD, obtains target data for pass one. The second, SGD, obtains target data for passes two and beyond. The third segment, STAL, principally routines STALL, WAD and WADOUT, allocates to targets without terminal ballistic missile defenses. The fourth segment, DEFAL, principally routines DEFALOC and RESVAL handles ballistic missile defended targets.

3.4.1 Overlay ALCINT. The routines in this overlay are straightforward and need little explanation beyond that below. However, the routine DATGRP places information on a sequential file (file code 23) which is used by the FGD segment. Table 3 shows the format and content of this file.

3.4.2 Overlay ALCMUL. The design of the weapon-to-target allocator utilizes a hierarchy of subroutines operating at different levels of detail. Figure 11 illustrates this hierarchy. The major functions associated with these subroutines are summarized below and related to the overall concept in subsequent paragraphs.

Subroutine MULCON is the first subroutine in the hierarchy and is responsible for the control and adjustment of the Lagrange multipliers. MULCON monitors the rate at which various classes and types of weapons are being allocated to the target system and makes appropriate adjustments in the values of the Lagrange multipliers. In this role, MULCON does not need any detailed information concerning actual allocation. It is concerned only with the actual rate of allocation of the available inventory as the targets are processed. To obtain the assignment of weapons to each successive target, MULCON simply calls subroutine STALL (Single Target Allocator) for targets without missile defenses, or subroutines STALL and DEFALOC if the target is defended. STALL and DEFALOC utilize the current values of the multipliers to make an allocation to the next target, then return control to MULCON.

The data acquisition for the allocation process is performed by the FRSTGD routine on pass one and SCNDGD on all other passes. Each of these routines brings in the proper records for the next target and prepares the weapon data for that particular target. The Weapon/Target Data File is read (or on the first pass in the RECALC mode calculated). On the first pass, FRSTGD then creates a record on file code 21 which, principally, contains the INACTIVE array (see table 4). If the user has requested range modification, FRSTGD may also write a record onto file code 22 in the same format as file code 15 (table 2). This new file serves as a source for replacement record for file code 15. SCNDGD reads each of these files in order to obtain the appropriate information.

Table 3. Weapon Data File from DATGRP

WORD	DESCRIPTION
1	Group type index number
2-4	Logical flag restrictions (99 flags packed 1 bit per flag)
5-9	Logical country location restriction (150 switches corresponding to codes in block CNCLS, packed 1 bit per switch)
10	Switch -- true if group is restricted MIRV
11	Logical MIRV restriction switches (packed 1 bit per switch)
12	Range multiplier
13	Refueled range multiplier
14	Minimum range replacement value
15	NALTDLY for group
16	ALTDLY for group
17	GLAT for group
18	GLONG for group
19	GREFCODE for bomber group, 0 for missile group
20	GYIELD for group
21	RANGE for group
22	CEP for bomber group, CEP at 0 range for missile group (CEP = RANGE * (CEP-CEPMIN)/(RANGE-RANGMIN))
23	SPEED for group
24	RANGED for group
25	RANGER for group (in DATAMAKE mode, PENPROB)
26	RNGMIN for group
27	GREFTIME for group
28	TOFMIN for group
29	CMISS for group
30	Slope of CEP equation for missiles, 0 for bombers

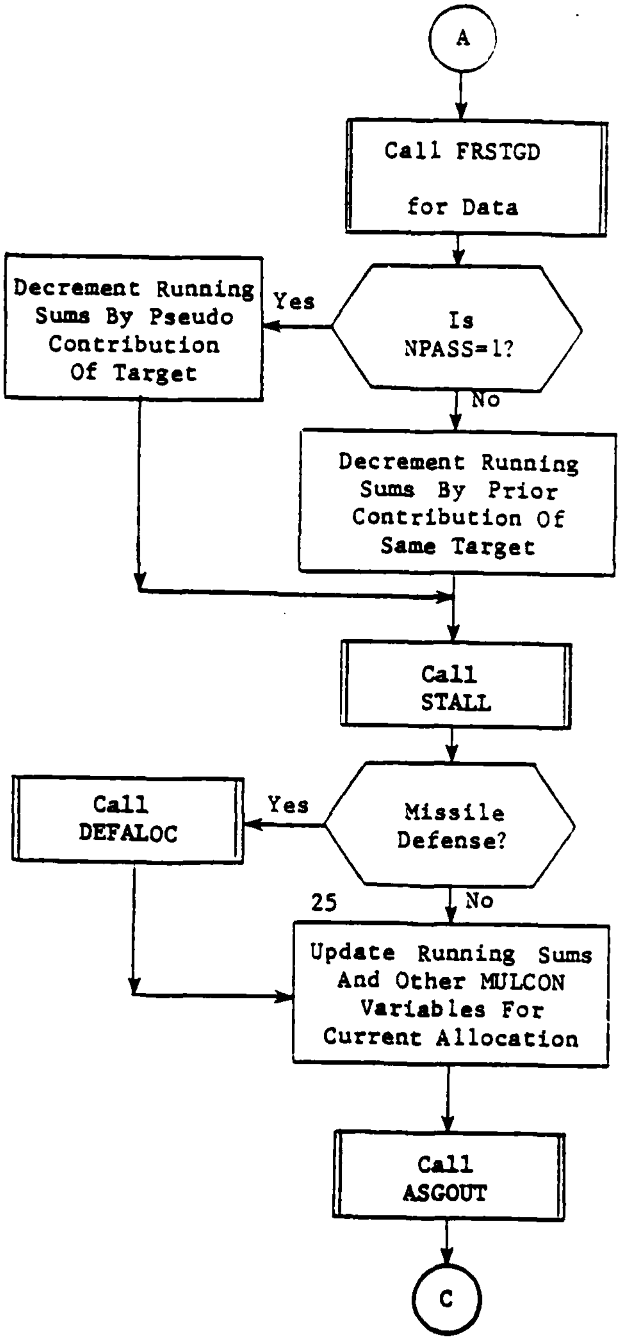

A
Call FRSTGD for Data
Is NPASS=1?
Yes
Decrement Running Sums By Pseudo Contribution Of Target
No
Decrement Running Sums By Prior Contribution Of Same Target
Call STALL
Missile Defense?
Yes
Call DEFALOC
No
25
Update Running Sums And Other MULCON Variables For Current Allocation
Call ASGOUT
C

Figure 12. Subroutine Mulcon
Part I: Bookkeeping Loop
(Part 1 of 2)

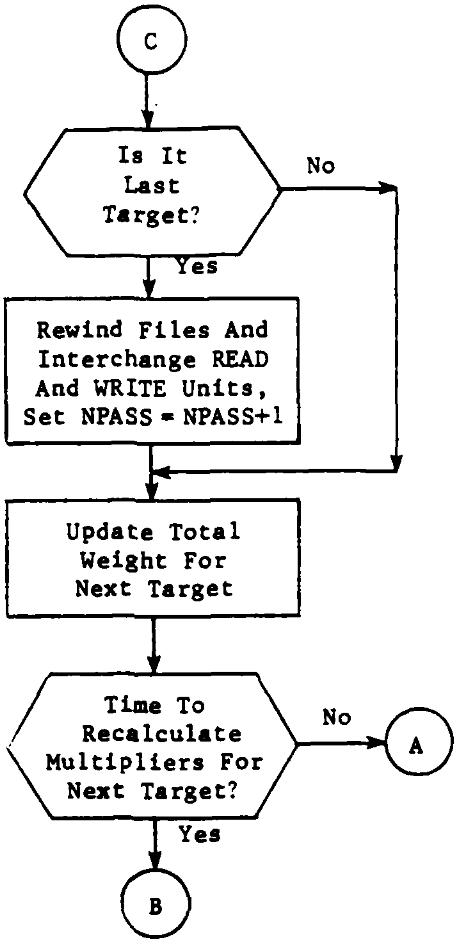

C
Is It Last Target?
No
Yes
Rewind Files And Interchange READ And WRITE Units, Set NPASS = NPASS+1
Update Total Weight For Next Target
Time To Recalculate Multipliers For Next Target?
No
A
Yes
B

Figure 12. (Part 2 of 2)

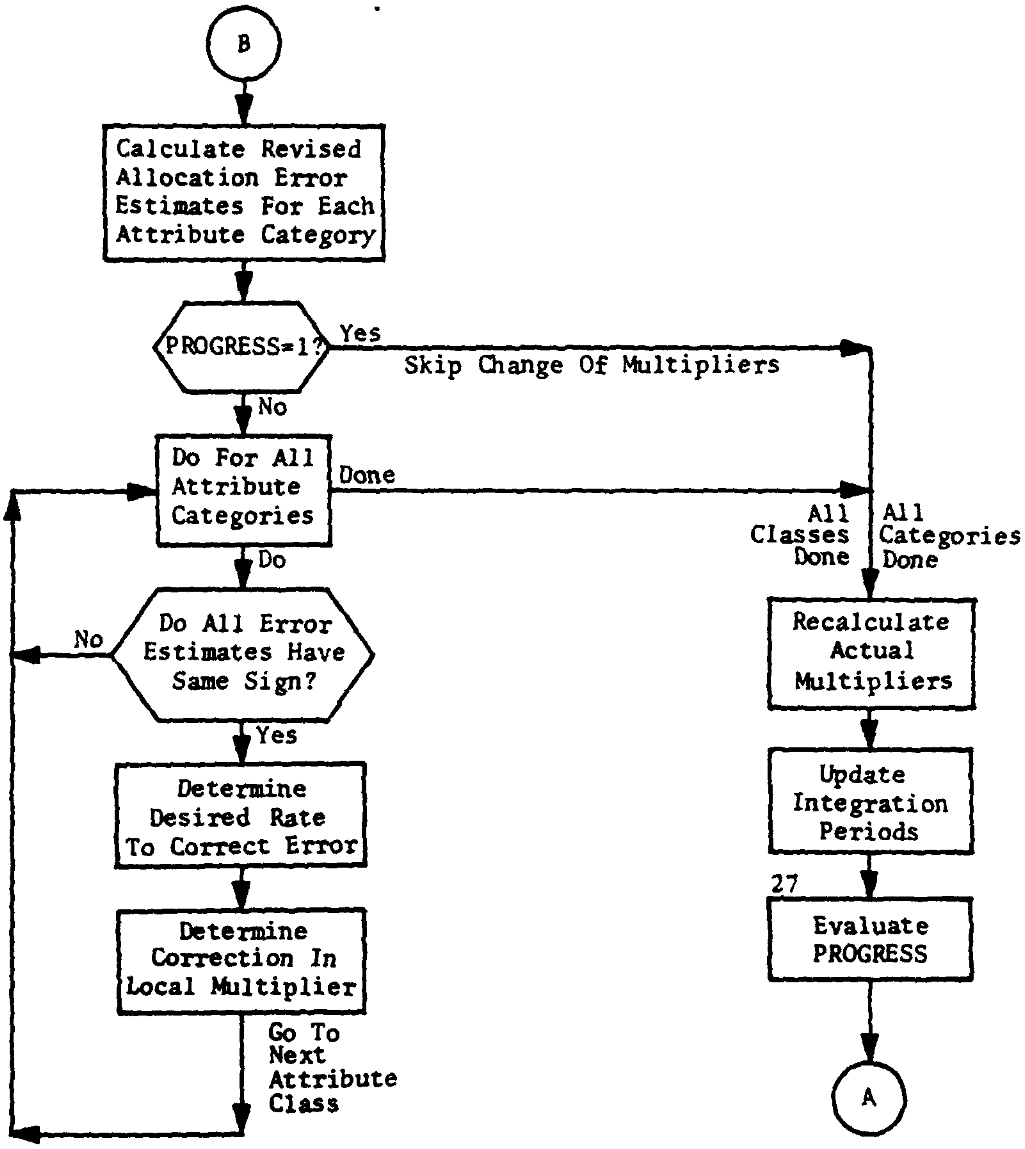

B
Calculate Revised Allocation Error Estimates For Each Attribute Category
PROGRESS=1?
Yes
Skip Change Of Multipliers
No
Do For All Attribute Categories
Done
Do
Do All Error Estimates Have Same Sign?
No
Yes
Determine Desired Rate To Correct Error
Determine Correction In Local Multiplier
Go To Next Attribute Class
All Classes Done
All Categories Done
Recalculate Actual Multipliers
Update Integration Periods
27
Evaluate PROGRESS
A

Figure 12. Part II: Computation Loop

Again, after the new allocation has been made, it is added into the running sums. Before passing on to the next target, the current value of the target weight is revised.

After every two to four targets, the Lagrange multipliers are updated by transferring control to the multiplier computation loop shown on figure 12. The error in the rate of allocation for each collection of weapons J is estimated. Three separate estimates are made corresponding to differing rates of increase of the target weights. If all estimates have the same sign, then a small adjustment of the multiplier in the indicated direction is made.

The revised local multipliers are then used to recalculate the Lagrange multipliers. During the closing phase (PROGRESS = 1), the local multipliers are not changed so the actual multipliers remain unchanged. The rate of change of the target weight is adjusted depending on the apparent size of the current error in the allocation rates.

Finally the progress of the allocation is evaluated and flags are set if the mode of operation is to change.

3.4.2.2 Subroutine STALL. STALL is basically a very simple routine. It is not responsible for computing payoffs. The payoffs are computed by WAD and summarized for STALL by WADOUT. Thus, the input to STALL consists only of the summary data provided by WADOUT. These data consist of only the following variables:

a. PPMX the maximum potential profit available if a weapon is added, and IPPMX the index to that weapon

b. PVRMX the maximum efficiency for any potential weapon that could be added, and IPVRMX the index to that weapon

c. DPMN the minimum differential profit produced by any weapon on the target, and IDPMN the index to that weapon.

Actually, of course, the profits and efficiencies mentioned above are based on a modified payoff (or BENEFIT) computed by WADOUT, which includes the actual payoff from WAD together with the premiums for staying close to the desired allocation rates. Thus, throughout the allocation, the operation of STALL remains the same; it simply tries to maximize this modified payoff. Changes in the mode of the allocation are thus accomplished in WADOUT simply by changing the way the payoffs are modified, so no change in the logic of STALL is required.

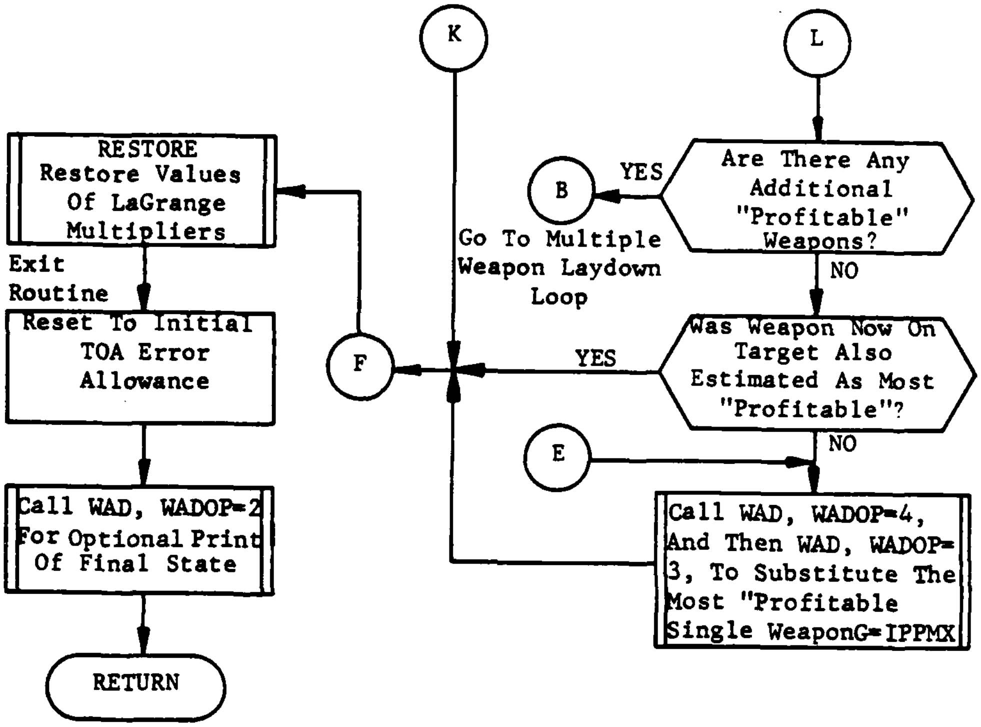


Figure 13. Part I
(Part 2 of 2)

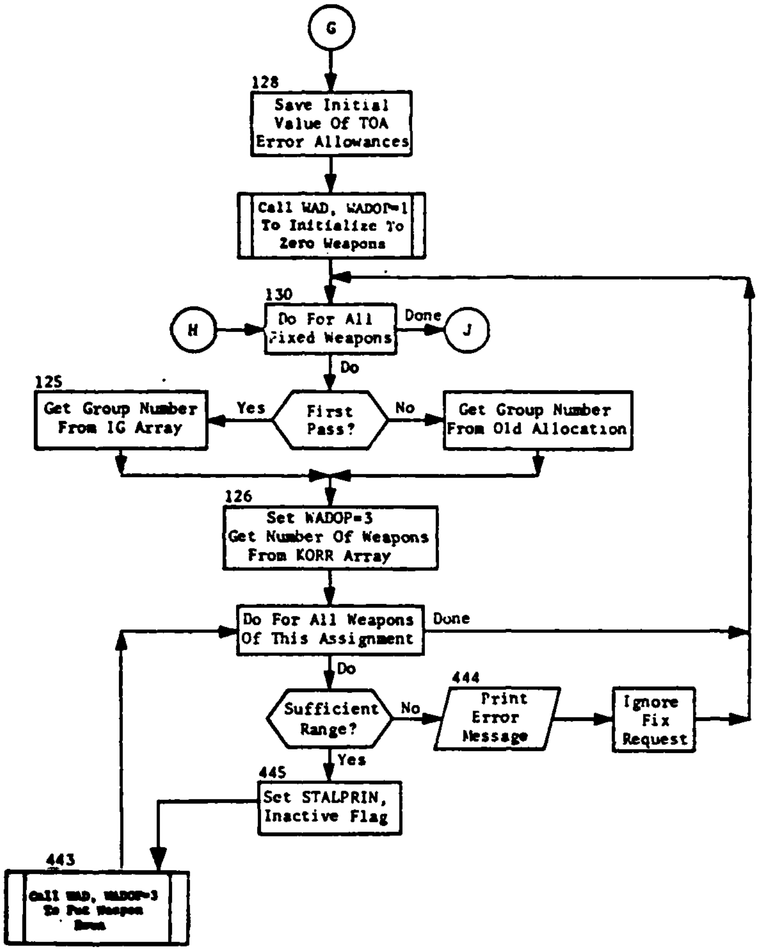

G
128
Save Initial Value Of TOA Error Allowances
Call WAD, WADOP=1 To Initialize To Zero Weapons
130
H
Do For All Fixed Weapons
Done
J
Do
125
Get Group Number From IG Array
Yes
First Pass?
No
Get Group Number From Old Allocation
126
Set WADOP=3 Get Number Of Weapons From KORR Array
Do For All Weapons Of This Assignment
Done
Do
Sufficient Range?
No
444
Print Error Message
Ignore Fix Request
Yes
445
Set STALPRIN, Inactive Flag
443
Call WAD, WADOP=3 To Put Weapon Down

Figure 13. Part II: Fixed Weapon Assignment Processing

If the IMATCH parameter is used, VTMIN and VTMAX retain their default values until the 0th order calculation VTZO indicates that MINKILL or MAXKILL have been reached. Then VTMIN or VTMAX is set correspondingly and thereafter operates as usual.

In any case, WADOUT modifies its calculations so that every weapon placed on target destroys at least that percentage of original target value specified by the user-input parameter MINDAMAG.

3.4.2.5 Subroutine PREMIUMS. The purpose of this subroutine is to compute the payoff premium required by STALL to avoid unduly large deviations from the desired allocation rate. During the closing phase, the premium is also used to cause STALL to close in to an allocation that exactly meets the stockpile constraints. PREMIUMS is called with one parameter which specifies for which group a new evaluation of the premiums (PREMIUM(G) and DPREMIUM(G)) is desired. The call results in the replacement of the old value of these premiums by a new value.

3.5 Common Block Definitions

The common blocks internal to the ALOC module are shown in table 5.

Table 5. ALOC Module Common Blocks (Part 1 of 10)

BLOCK	ARRAY OR VARIABLE	DESCRIPTION
ALERUN	ALERREST(360,3)	Estimate of allocation errors
	RUNSUM(360,3)	Running sum of target weight times weapons allocated
C33	NBLN	Number of ballistic interceptors
	XXYY	Not used
	VT	Target value remaining
	TGTWT(3)	Weighting values
	PAYOFF	Target value destroyed
	COST	Sum of weapon values allocated
	PROFIT	PAYOFF - COST
	DPROFIT	Change in PROFIT from last pass
	WRTEST	Test parameter
CNCLS	CNTRY(100)	List of country codes
	CLSS(20)	List of target classes
	NCNT	Number of country codes
CONTRO	WADOP	WAD option
	PROGRESS	Measure of allocation progress
	NPASS	Allocation pass
CORSTF	COSCOR(30)	Cosine of corridors orientation point latitude
	DPLAT(30)	Difference between latitude of orientation and origin
	DPLONG(30)	Difference between longitude of orientation and origin

Table 5. (Part 1 of 10)

BLOCK	ARRAY OR VARIABLE	DESCRIPTION
CORSTF (cont.)	CORLN(30)	Distance from orientation to origin
	TDEFDIST(30)	Total precorridor distance
	TATTRPRE(30)	Total precorridor attrition
	ATTROCOZ(30)	Corridor attrition rate
	ATTRSUZ(30)	Corridor suppression rate
	HILOATZ(30)	Ratio of high to low attrition
	CRLENGTH(30)	Corridor length

Table 5. (Part 2 of 10)

BLOCK	ARRAY OR VARIABLE	DESCRIPTION
CORSTF (cont.)	ORGLAT(30)	Origin latitude
	ORGLONG(30)	Origin longitude
	DISTCDZ(30)	Distance from origin to target
	ATTRAD(30)	Sum of precorridor and corridor attrition
	CROSSDST(30)	Perpendicular distance from axis to target
	ENTLAT(30)	Entry latitude
	ENTLONG(30)	Entry longitude
	DISTDG	Distance from target to recovery base
	MAXCOR	Number of corridors
CURSUM	CSALL	Number of targets assigned to all weapons
	CSREG(5)	Number of targets assigned to weapons from a given region
	CSCLAS(2)	Number of targets assigned to weapons from a given class
	CSTYPE(100)	Number of targets assigned to weapons of a given type
	CSGRP(250)	Number of targets assigned to weapons from a given group
	CSOTH(2)	Number of targets assigned to weapons with a given alert status
DEFCOM	RATM	Highest return rate from DEFALOC
	ISALFX(250)	Storage for fixed salvo numbers
	NSL(250)	Number of salvoed weapons available
	RATE(250)	Rate of return for missile on defended target
DEFRES	NOWEP(250)	Number of weapons assigned by DEFALOC
	VTDX	Surviving target value
	NTX(3)	Terminal defender estimates
	PX(3)	Probability of NTX

Table 5. (Part 3 of 10)

BLOCK	ARRAY OR VARIABLE	DESCRIPTION
DYNAMIC		Contains allocation
	IG(30)	Group assigned
	KORRX(30)	Corridor assigned
	RVALX(30)	Relative value of assignment
	PENX(30)	Penetration probability
	TOARR(30)	Time of arrival
	ISAL(30)	Salvo number
	NUMFIX	Number of fixed assignments
	NUM	Number of assignments
FIRST	FIRST	Indicates first target of pass
	END	Indicates end of target list
	F22LSW	Indicates file 22 in use
FLGSTF	LFLAG(688)	Packed logical flag restrictions
	LCNTRY(695)	Packed logical location restrictions
	NMMRV	Number of restricted MIRV groups
	MRNAM(100)	Payload table name of restricted MIRV
	LCLAS(64)	Packed logical MIRV class restrictions
	RNMUL(250)	Range multiplier
	RNRMUL(250)	Refueled range multiplier
	RNMIN(250)	Range minimum replacement
FORMTT	INWORD	Value which needs a format
	NFORMAT	Format for INWORD
GRPHDR	IWGHDR	IDS Reference Code for weapon group header

Table 5. (Part 4 of 10)

BLOCK	ARRAY OR VARIABLE	DESCRIPTION
GRPSTF		Contains record from file 25
	ITYPE	Group's type index
	LFLAG (3)	Flag restrictions
	LCNTRY(3)	Location restrictions
	LMRSW	Indicate if group is restricted MIRV
	LCLASS (1)	MIRV restrictions
	RMUL	Range multiplier
	RRMUL	Refueled range multiplier
	RMIN	Minimum range replacement
	GPARAM (16)	Contains the following group parameters in order: NALTDLY, ALTDLY, GLAT, GLONG, GREFCODE, GYIELD, RANGE, CEP, SPEED, RANGED, RANGER, RNGMIN, GREFTIME, TOFMIN, CMISS, SLOPE
INITSW	RECALC	Indicates RECALC mode
	PUNSW	Indicates output of final lambdas is desired
	FLAGSW	Indicates flag restrictions
	LOCRSW	Indicates location restrictions
	RMODSW	Indicates range modifications
	MRVRSW	Indicates MIRV restrictions
	PUNIT	Logical unit on which final lambdas are to be output
IRTLEN	LENI(4)	Filecode 15 lengths
	KR	Defense switch indicator
	NSTK2	Switch which indicates if more data is needed

Table 5. (Part 4 of 10)

BLOCK	ARRAY OR VARIABLE	DESCRIPTION
KASG	KGROUP(8)	Group
	XALAT	Target latitude
	XALONG	Target longitude
	KXORR(15)	Corridor assignment
	XRVAL(30)	Value of target
	XPEN(30)	Penetration probability
	XARV(30)	Time of arrivel
	KSALVO(6)	Salvo number
	KHOB(1)	Height of burst indicator
	KFXD(1)	Fixed target indicator
LACB	LALL	Lambda for all weapons
	LAREG(5)	Lambda for a given region
	LACLAS(2)	Lambda for a given class
	LATYPE(100)	Lambda for a given weapon type
	LAGRP(250)	Lambda for a given group
	LAOTH(2)	Lambda for a given alert status
LGR	LGREF(250)	Local Group Reference number

Table 5. (Part 5 of 10)

BLOCK	ARRAY OR VARIABLE	DESCRIPTION
NALLY	NALL(250)	Number from group allocated on this pass
	RNALL(250)	Number from group currently allocated
NOWPS	NOALL	Number of weapons total
	NOREG(5)	Number of weapons in a given region
	NOCLAS(2)	Number of weapons in a given class
	NOTYPE(100)	Number of weapons of a given type
	NOGRP(250)	Number of weapons in a given group
	NOOTH(2)	Number of weapons with a given alert status
PAYOFF	OPROFIT	Profit from old allocation
	SPAYOFF	Sum of all payoffs
	SUMCOST	Sum of all costs
	SPROFIT	Sum of all profits
PAYSAV	GSCC(100)	CCREL for payload
	GSREL(100)	REL for payload
	GSEASM(100)	EXPASM for payload
	GSLINT(100)	LCHINT for payload
	NGSWHD(100)	NWHDS for payload
	NGSDEC(100)	DECOYS for payload
	GPAYALT(100)	PAYALT for payload
	GYLDASM(100)	ASM yield for payload
	IWHOB(100)	Payload height of burst 0 = for ground 1 = for air 2 = if not preset

Table 5. (Part 6 of 10)

BLOCK	ARRAY OR VARIABLE	DESCRIPTION
PNAV	GSPKNAV(100)	PKNAV for payload
PREMS	PREMIUM(250)	Premium for using weapon
	DPREMIUM(250)	Premium for deleting weapon
	SUMPREM	Sum of premiums
	TBENEFIT	Total benefit
PREMSV	PRSV(2)	Parameter for premiums subroutine
PRNTCN	IDO(40)	Print request activation switch
	INDEXPR(40)	Print request selection number
	JPASS(40)	Print request first pass
	JTGTP(40)	Print request first target
	LPASS(40)	Print request last pass
	LTGT(40)	Print request last target
	KTGTFREQ(40)	Target print frequency
	ICOUNT(40)	Print request frequency counter
	MAXREQ	Maximum number of requests
	MPRNT	Number of array entries
	NREQ	Number of requests
PRTMUL	DELTEFF	Increase in profit/VALWPNS
	SDELTEFF	Sum of DELTEFF
	VALWPNS	Sum of all weapon values (lambdas)
	VALERR	Value of surplus plus deficit weapons
QTKCOM	QTK	First PK
	QTK	Second PK
REFPNT	RFLAT(10)	Refuel point latitude
	RFLONG(10)	Refuel point longitude
SALVO	NSALW	Number of salvoed weapon groups
	MXSAL(75)	Maximum salvo number per weapon group

Table 5. (Part 6 of 10)

SALVO (cont.)	NSALAL(450)	Running sums of salvo allocation (six words per salvoed group - packed four sums per word)
	LXIHAVE(50)	Packed logical switch indicating salvo with weapons

Table 5. (Part 7 of 10)

BLOCK	ARRAY OR VARIABLE	DESCRIPTION
SALVO (cont.)	SAVLAM(250)*	Storage for salvoed weapon lambdas (contains average payload difference for bombers - AVDE)
	MYSAL(250)*	Available salvo (contains bomb/ASM selection for bombers - ISETPAY)
	P(250)*	Balance parameter (contains current utilization of ASMs for bomber - FASM)
	ISALPT(250)	Salvo index, indexes arrays MXSAL, NSALAL and LXIHAVE: 0 for nonsalvoed groups
SMATAD	SMNOMIRV(3)	SMAT parameters for non-MIRVs
	SMATMIRV(3)	SMAT parameters for MIRVs
SURPW	SURPWP(250)	Estimated weapon surplus
TABLE	TABLE(101)	Table of square root law K-factors
TGTSAV	TGTLAT	Target latitude
	TGTLONG	Target longitude
	TGTCLS	Target class name
	VO(2)	Target value per hardness component
WADFIN	VTP(250)	Value remaining at target after weapon added
	DELVT(30)	Difference in surviving value
	NUMO	Numbers of old allocations
	IGO(30)	Group numbers of old allocation

* For bomber groups these arrays are equivalenced to arrays AVDE, ISETPAY and FASM.

Table 5. (Part 8 of 10)

BLOCK	ARRAY OR VARIABLE	DESCRIPTION
WADFIN (cont.)	IOP	Number of adds and deletes on this target
	IOPS	Sum of IOP
WADLOC	NWP(10)	Number of weapons in TOA set
	VALQ(10)	Unattritioned value at TOA
	MU(10,2)	Sum of means through TOA set
	SIG(10,2)	Sum of variance through TOA set
	V(11,2)	Unattritioned component value
	S(10,2)	Component survival probability through TOA set
	VS(10,2)	=(V(N,JH)-V(N+1,JH) * S(N,JH)
	VSN(11,2)	= VSN(N-1,JH) + VS(N-1,JH)
	ITOA(250)	TOA index for group
	IADDTOA(250)	1 if new TOA set required
	SIGP(250,10,2)	Increase in variance for TOA set if weapon added
	DSIG(250,2)	Temporary contribution of weapon
	SIGD(30,10,2)	Change in variance if weapon deleted
WADOTX	PVRMX	Maximum efficiency
	IPVRMX	Index of weapon achieving PVRMX
	PPMX	Maximum profit
	IPPMX	Index of weapon achieving PPMX
	DPMN	Minimum profit
	IDPMN	Index of weapon achieving DPMN
	NUMMAX	Maximum number of weapons allowed per target
	NW	Number on target
	TPMX	Largest potential profit
	NTOA	Number of TOA sets
	NOTAMAX	Maximum TOA sets
	VTMIN	Lower target destruction minimum

Table 5. (Part 9 of 10)

BLOCK	ARRAY OR VARIABLE	DESCRIPTION
WADOTX (cont.)	VTMAX	Maximum acceptable surviving target value
	ALPHA	Factor on value required to justify VTMAX
	VTEF	Maximum of target value remaining and VTMIN
	VTZO	Total surviving target value
	VTO	First target component value
	STALPRIN	Stall print code
	G	Group Number
	N	Allocation number
WADWPN	INACTIVE(250)	Active group switch (0 = Active)
	TOA(250)	Time of arrival on target
	TVALTOA(250)	Value of target at arrival time
	VTOA(250,2)	Value per component at TOA
	MUP(250,2)	Contribution of weapon to mean if added
	RISK(6,250,2)	Relative risk of weapon interaction
	SSIG(250,2)	Square root of ln of SSKP
	MORR(250)	Optimal corridor
	PEX(250)	Penetration probability
	XMUP(250,2)	MUP for alternate warhead - (ASM for bomber group, single warhead for MRV)
	ILAW	Damage law in use
WEPSAV	IP(250)	Group payload index
	GSSBL(250)	Group SBL
	IGTYP(250)	Group type index
	IGLERT(250)	Group alert status index
	IGREG(250)	Group region index
WPFIX	NWPNS(250)	Number of weapons in group

Table 5. (Part 10 of 10)

BLOCK	ARRAY OR VARIABLE	DESCRIPTION
WPFIX (cont.)	NMTYP(100)	Type names
	LAM(250)	Group lambda
	ITCL(100)	Weapon type class index (1=missile, 2=bomber)
WTS	WTFAC(3)	Divide into old weight for commeasurability
	WTRATE(3)	Rate of increase of weights
	WTSUM(3)	Sum of weights
XFPX	PENALT(30)	Penetration probability for corridor

3.6 Subroutine ENTMOD

PURPOSE:	Entry module for ALOC
ENTRY POINTS:	ENTMOD (first subroutine called when overlay ALOC is executed)
FORMAL PARAMETERS:	None
COMMON BLOCKS:	C30, CNCLS, GRPHDR, INITSW, LACB, NOWPS, PAYSAV, PRNTCN, SALVO, SMATAD, TABLE, WEPSAV, WPFIX, OOPS, IRTLEN
SUBROUTINES CALLED:	INITIAL, MULCON, RANSIZ
CALLED BY:	MODGET

Method:

First RANSIZ is called to set the record size of file 25. Next the ALCINT overlay is read in and executed. If no input error has been detected, the ALCMUL overlay is executed. Finally, the final multipliers are written (or punched) if the user has so directed.

Subroutine ENTMOD is illustrated in figure 15.

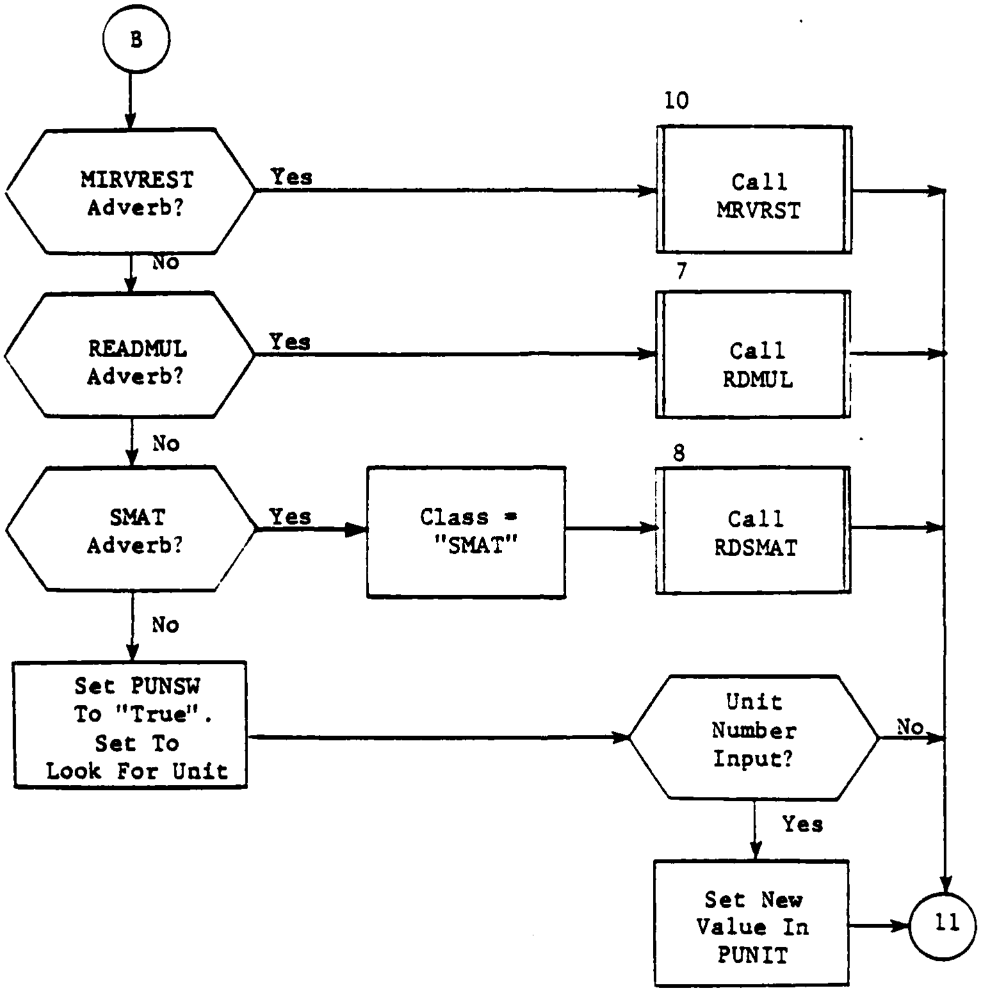


Figure 16. (Part 3 of 4)

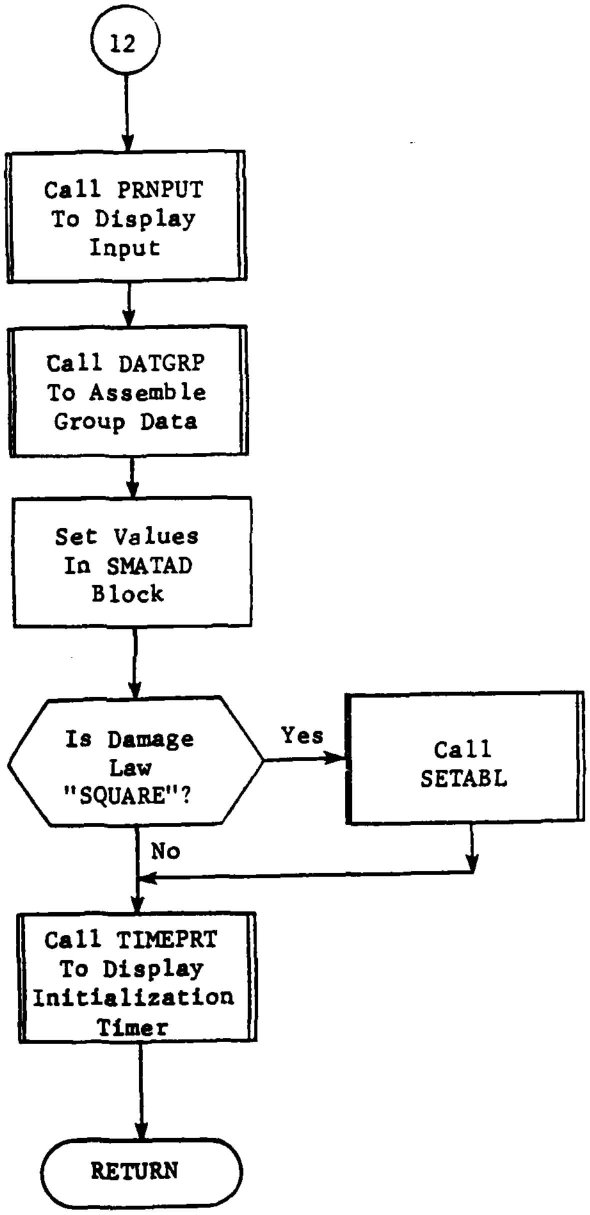

12
Call PRNPUT To Display Input
Call DATGRP To Assemble Group Data
Set Values In SMATAD Block
Is Damage Law "SQUARE"?
Yes
Call SETABL
No
Call TIMEPRT To Display Initialization Timer
RETURN

Figure 16. (Part 4 of 4)

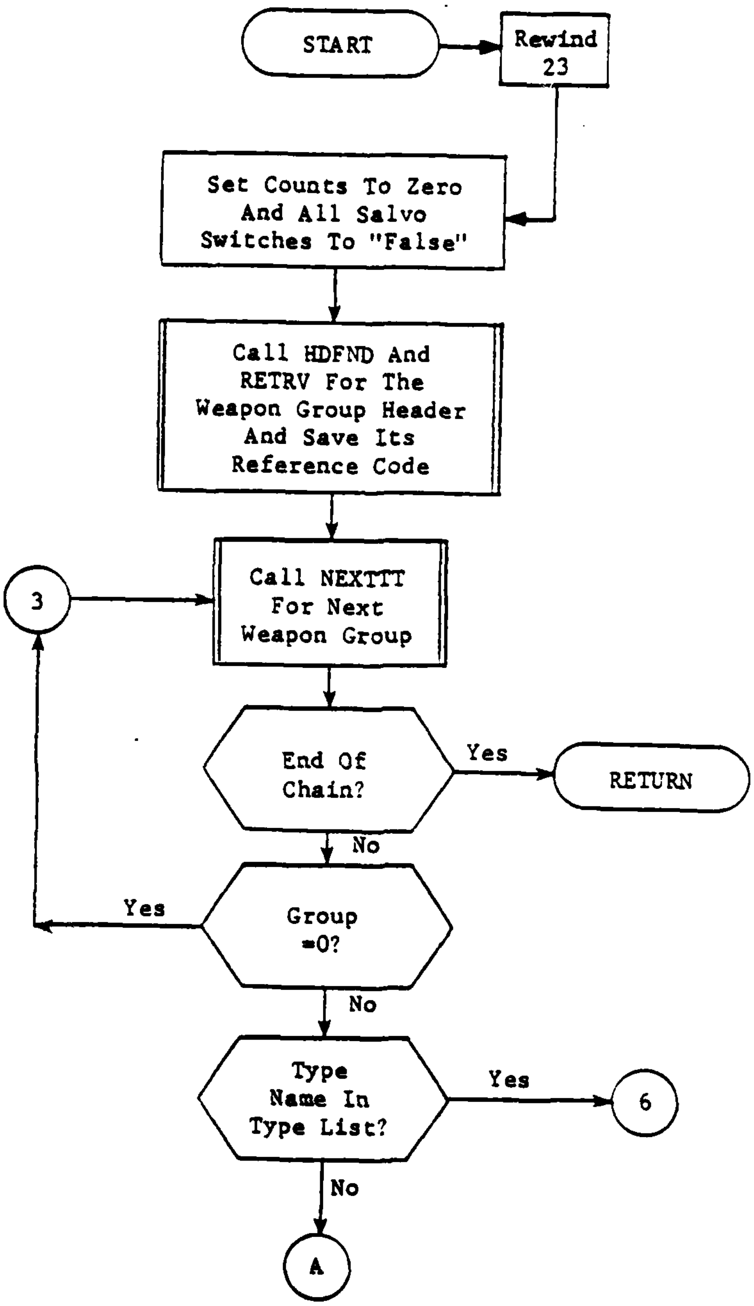

START
Rewind 23
Set Counts To Zero And All Salvo Switches To "False"
Call HDFND And RETRV For The Weapon Group Header And Save Its Reference Code
3
Call NEXTTT For Next Weapon Group
End Of Chain?
Yes
RETURN
No
Yes
Group =0?
No
Type Name In Type List?
Yes
6
No
A

Figure 18. Subroutine DATGRP (Part 1 of 6)

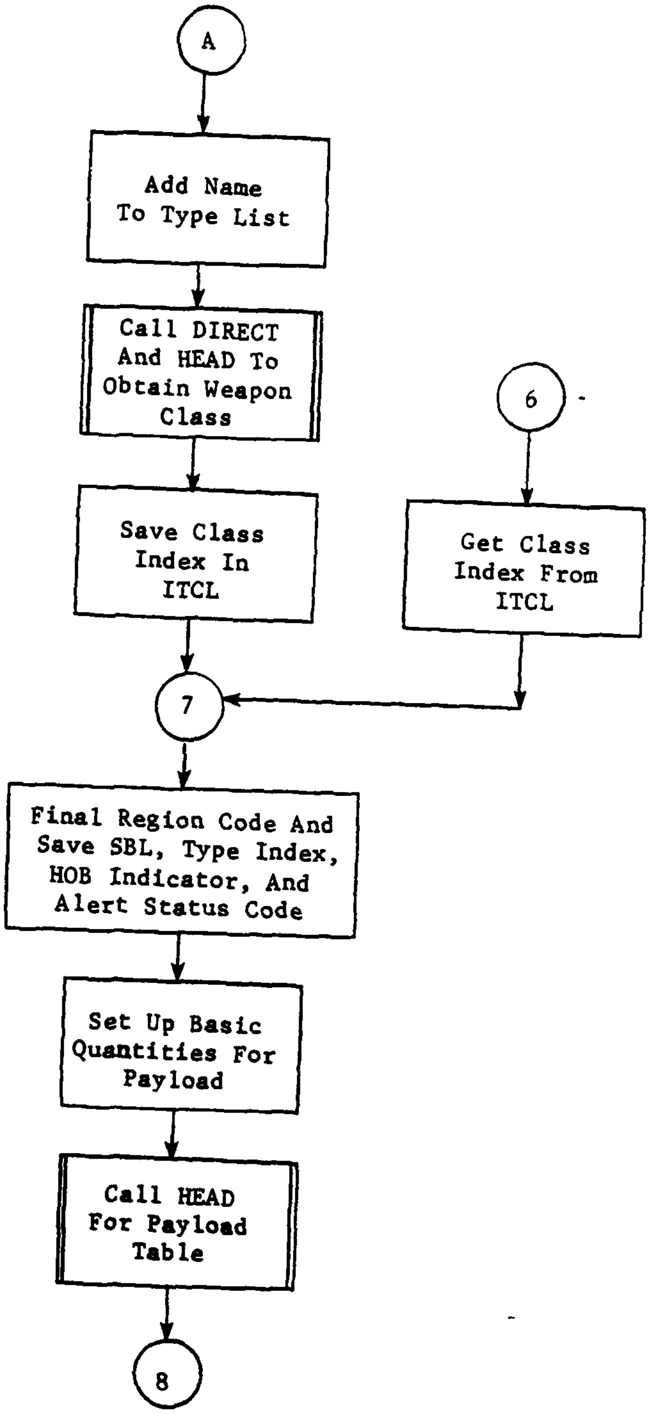
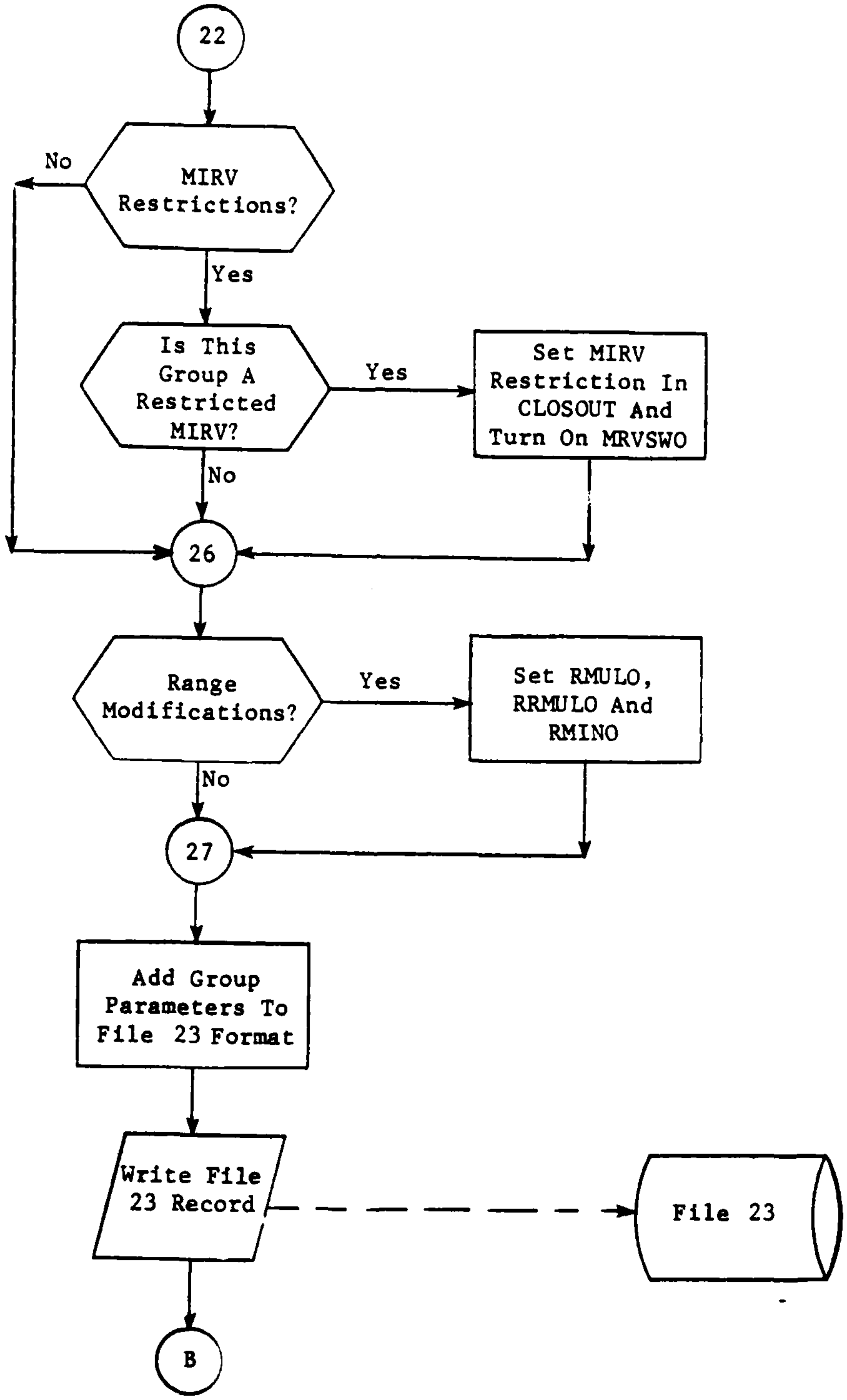

A
Add Name To Type List
Call DIRECT And HEAD To Obtain Weapon Class
6
Save Class Index In ITCL
Get Class Index From ITCL
7
Final Region Code And Save SBL, Type Index, HOB Indicator, And Alert Status Code
Set Up Basic Quantities For Payload
Call HEAD For Payload Table
8

Figure 18. (Part 2 of 6)


22
MIRV Restrictions?
No
Yes
Is This Group A Restricted MIRV?
Yes
Set MIRV Restriction In CLOSOUT And Turn On MRVSWO
No
26
Range Modifications?
Yes
Set RMULO, RRMULO And RMINO
No
27
Add Group Parameters To File 23 Format
Write File 23 Record
File 23
B

Fjgure 18. (Part 5 of 6)

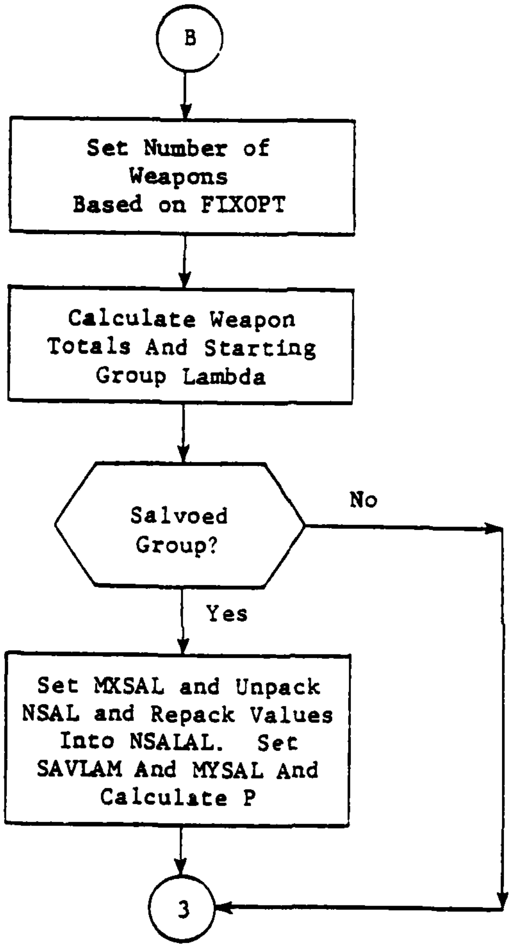

B
Set Number of Weapons Based on FIXOPT
Calculate Weapon Totals And Starting Group Lambda
Salvoed Group?
No
Yes
Set MXSAL and Unpack NSAL and Repack Values Into NSALAL. Set SAVLAM And MYSAL And Calculate P
3

Figure 18. (Part 6 of 6)

3.8 Subroutine MULCON*

PURPOSE: The primary purpose of MULCON is to adjust the Lagrange multipliers during the allocation. It also monitors the progress of the allocation, and controls the flow of the module throughout the allocation.

ENTRY POINTS: MULCON

FORMAL PARAMETERS: None

COMMON BLOCKS: ALERUN, C10, C30, C33, CONTRO, CORSTF, CURSUM, DYNAMI, FIRST, GRPHDR, LACB, NALLY, NOWPS, PAYOFF, PAYSAV, PREMS, PRTMUL, REFPNT, SALVO, SURPW, TARREF, TGTSAV, WADFIN, WADOTX, WADWPN, WEPSAV, WPFIX, WTS, LGR

SUBROUTINES CALLED: ADDSAL, ASGOUT, BOMPRM, DEFALOC, DIRECT, FRSTGD, LLINK, MODFY NEXTTT, PRNTALL, PRNTCON, PRNTNOW, SCNDGD, STALL, TIMEME, ASCSAV

CALLED BY: ENTMOD (ALOC)

Method:

MULCON proceeds by making multiple passes through the target list until the correct number of weapons have been allocated. During this process, the Lagrange multipliers are modified to force the allocation to converge to the given stockpile.

The flow of operations in MULCON is illustrated in figure 29. The first sheet represents the flow in five parts of the program; Part I, the initialization phase; Part II, first pass processing; Part III, the main flow; Part IV, processing after allocation; and Part V, multiplier adjustment. The flow diagrams are annotated with statement references to the FORTRAN listing and are in sufficient detail to be largely self-explanatory. In the following sections comments are made only where the flow diagrams require some explanation.

Part I: Initialization

The routine begins by calling utility subroutine TIMEME to initialize the clock. NPASS and ITGT are initialized so that the print control routine PRNTCON will know what prints to activate, and then PRNTCON is called to do so. A call for a summary print of starting multipliers (PRNTALL(11)) follows.

* First routine of overlay ALCMUL.

The sensitivity of the allocation errors to the value of the local multipliers is estimated. With a fairly large value for the linear premium PRM all sensitivities are about the same independent of the size of the group; thus, PARTIAL(J) is simply set to -1.0.

RUNSUM and WTSUM are initialized as if the starting pseudoallocation were exact, but the weight given to that allocation is reduced by the square root of the number of targets so that it does not take too long for data on actual allocation rates to produce a significant effect on the estimated rates.

Part II: First Pass Processing

This section deals with the target processing on the first pass. On this pass FRSTGD is called which processes the target, brings in the weapon data and, if necessary, calculates the Weapon/Target Data File. FRSTGD also reads in any fixed allocation. Finally, the pseudoallocation is removed from the running target weights.

Part III: Main Flow (After First Pass)

This section deals with the target processing on passes after the first. First a series of calls to PRNTALL performs intertarget prints. Next PRNTCON is called to reset print controls. Then, if this is not the first pass processing continues. SCNDGD is called to process target data, bring in the weapon related data and read in the old allocation. If PROGRESS is 2 the variable IVERIFY is checked. If this is a verification pass the limits are checked and if not achieved the pass continues. If all processing is complete PRNTNOW is called for final print. Then the weapon group chain is cycled and the attribute NUMALOC set equal to RNALL for each group.

However, if processing is to continue, the termination section is ignored and the program gets ready to generate a new allocation to replace the one just read. On the first pass the old allocation is a pseudo-allocation and the replacement is done elsewhere (see Part II). The replacement is accomplished by removing the contribution of the old allocation* to all running sums before the new allocation is generated. This can be done in this simple way because the values of COST, PAYOFF, PROFIT, TGTWT, and DPROFIT, then in memory, are those just read in from the TARCDE record and so they correspond to the old allocation. Since the quantities RUNSUM and WTSUM were divided by WTFAC at the end of the last pass, the old TGTWT must also be divided by this same factor to make it commensurate before it is subtracted out.

The reason that REVCOST must be computed is that the values of the multipliers have probably changed (unless PROGRESS = 1.0) since the prior

* The fixed weapons are ignored because they do not contribute to the running sums.

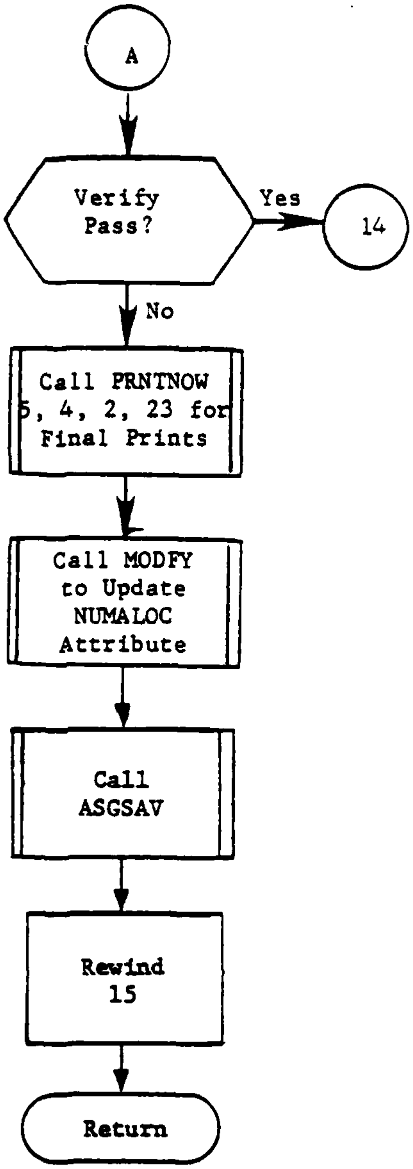

A
Verify Pass?
Yes
14
No
Call PRNTNOW 5, 4, 2, 23 for Final Prints
Call MODFY to Update NUMALOC Attribute
Call ASGSAV
Rewind 15
Return

Figure 29. Part III: (Part 2 of 3)

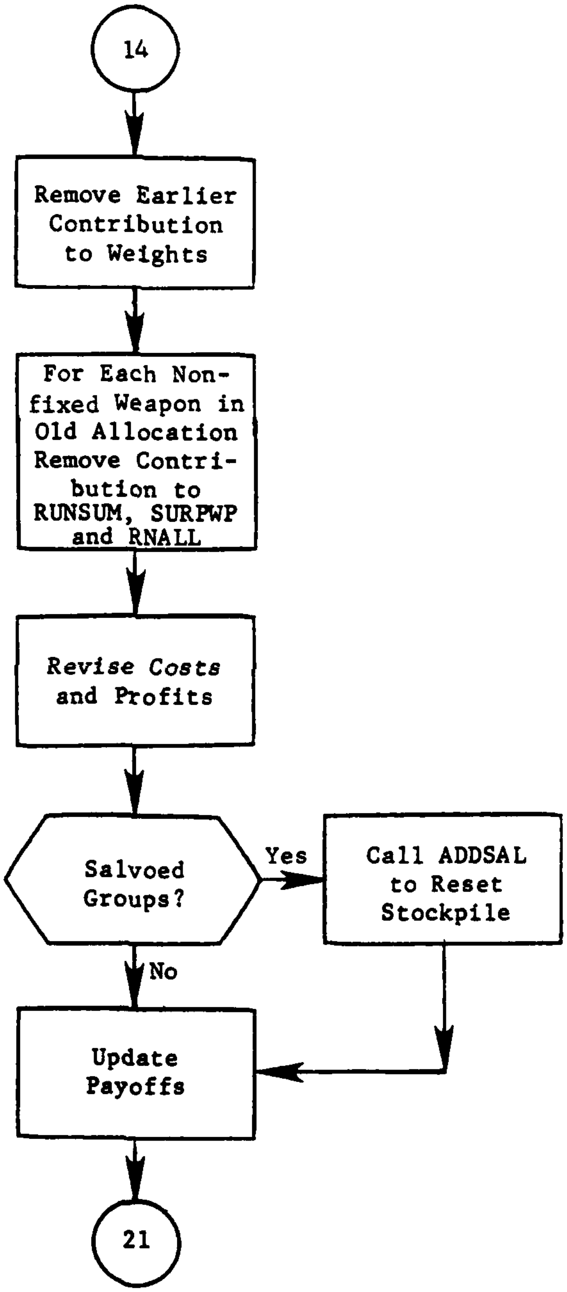

14
Remove Earlier Contribution to Weights
For Each Non-fixed Weapon in Old Allocation Remove Contribution to RUNSUM, SURPWP and RNALL
Revise Costs and Profits
Salvoed Groups?
Yes
Call ADDSAL to Reset Stockpile
No
Update Payoffs
21

Figure 29. Part III: (Part 3 of 3)

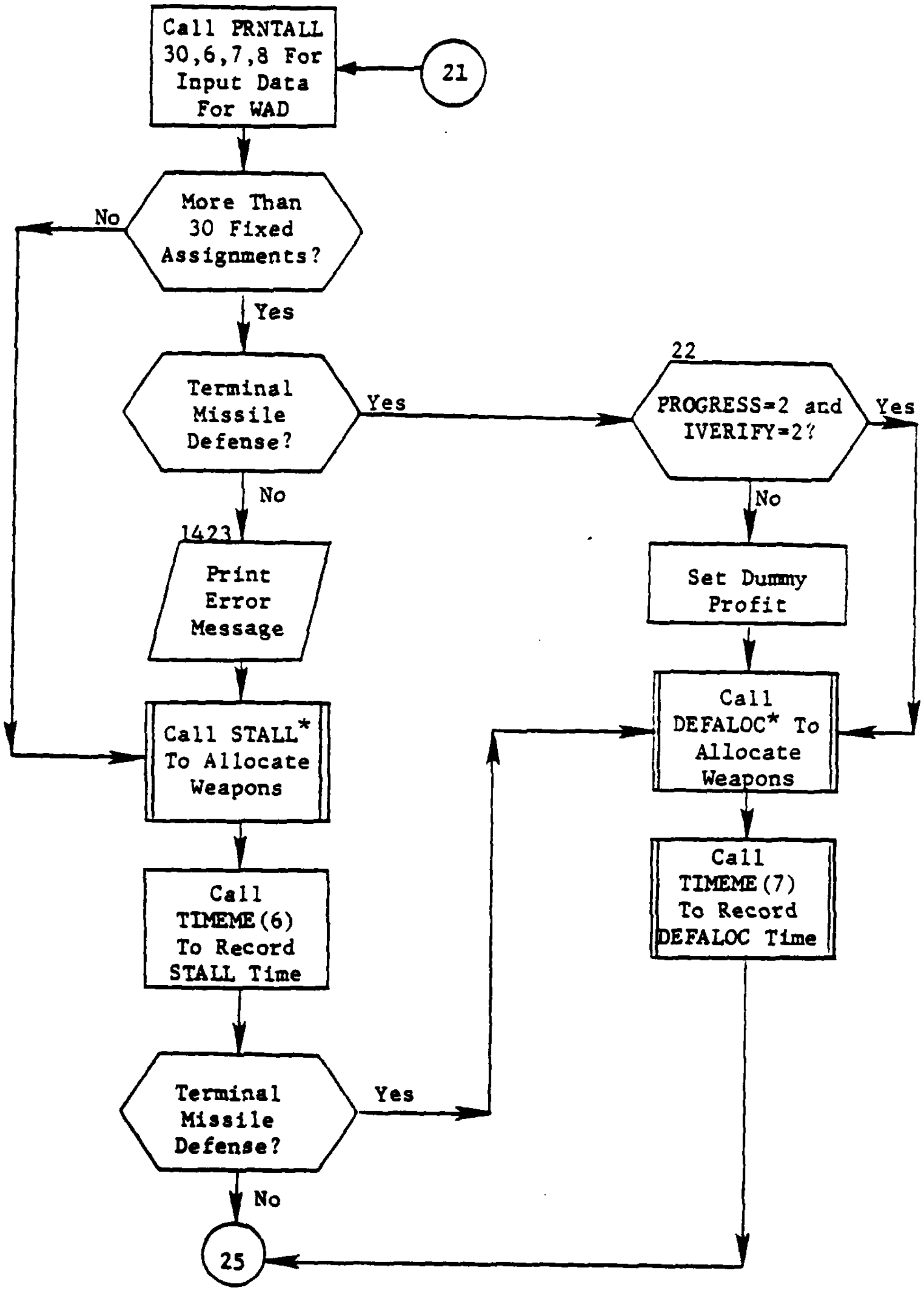

Call PRNTALL 30,6,7,8 For Input Data For WAD
21
More Than 30 Fixed Assignments?
No
Yes
Terminal Missile Defense?
Yes
No
1423
Print Error Message
Call STALL* To Allocate Weapons
Call TIMEME(6) To Record STALL Time
Terminal Missile Defense?
Yes
No
25
22
PROGRESS=2 and IVERIFY=2?
Yes
No
Set Dummy Profit
Call DEFALOC* To Allocate Weapons
Call TIMEME(7) To Record DEFALOC Time

Figure 29. Part IV: Processing After Allocation (Part 1 of 4)

*Called via FORTRAN routine LLINK

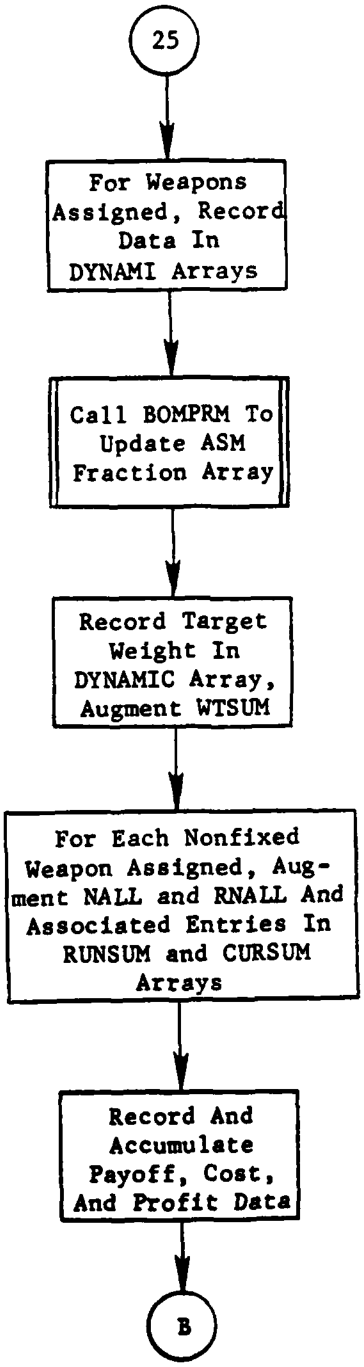

25
For Weapons Assigned, Record Data In DYNAMI Arrays
Call BOMPRM To Update ASM Fraction Array
Record Target Weight In DYNAMIC Array, Augment WTSUM
For Each Nonfixed Weapon Assigned, Augment NALL and RNALL And Associated Entries In RUNSUM and CURSUM Arrays
Record And Accumulate Payoff, Cost, And Profit Data
B

Figure 29. Part IV: (Part 2 of 4)

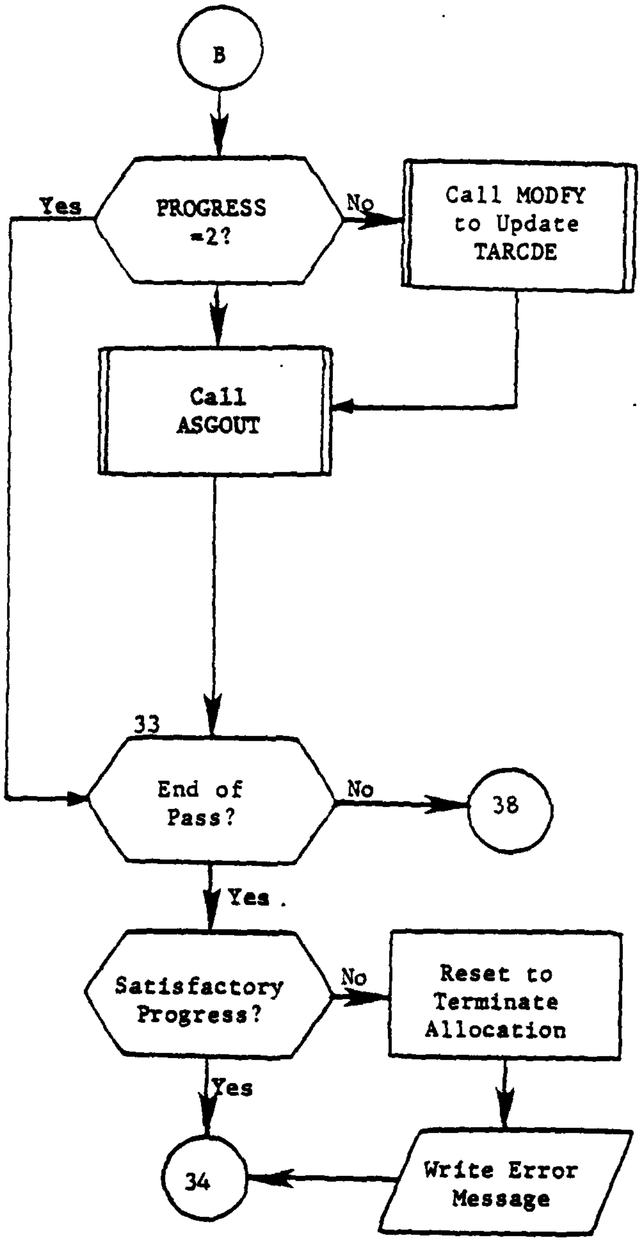


Figure 29. Part IV: (Part 3 of 4)

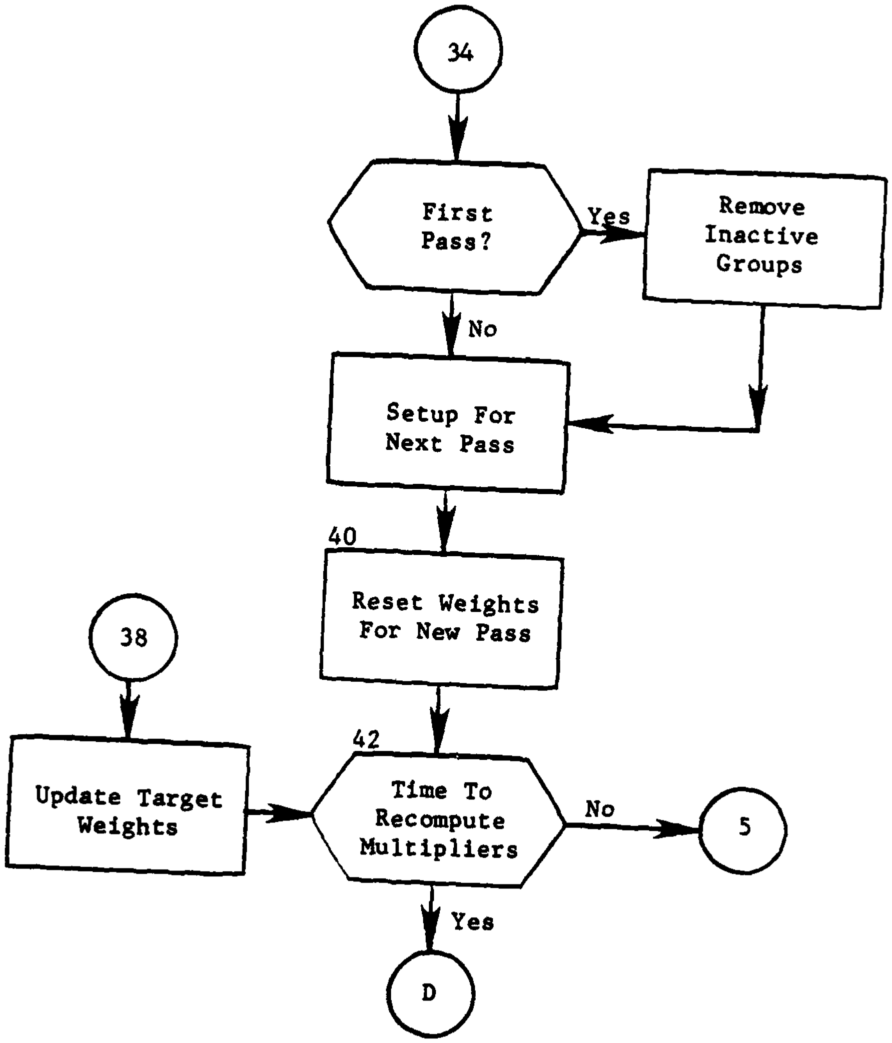


Figure 29. Part IV: (Part 4 of 4)

3.8.1 Subroutine ADDSAL

PURPOSE: This routine updates the stockpile for salvoed weapons

ENTRY POINTS: ADDSAL

FORMAL PARAMETERS:
IGP - group number
IOPT - option code
NUR - index to ISAL array
ISALIN - salvo number

COMMON BLOCKS: DYNAMI, SALVO

SUBROUTINES CALLED: None

CALLED BY: MULCON, STALL, WAD, DEFALOC

Method:

If the weapon is a nonsalvoed weapon but a bomber, ISAL is set to indicate whether weapon is a gravity bomb or ASM. Otherwise the variable IDIFF is set depending upon the option. From this the number in NSALAL (packed 4 per word) is either incremented or decremented.

Subroutine ADDSAL is illustrated in figure 30.

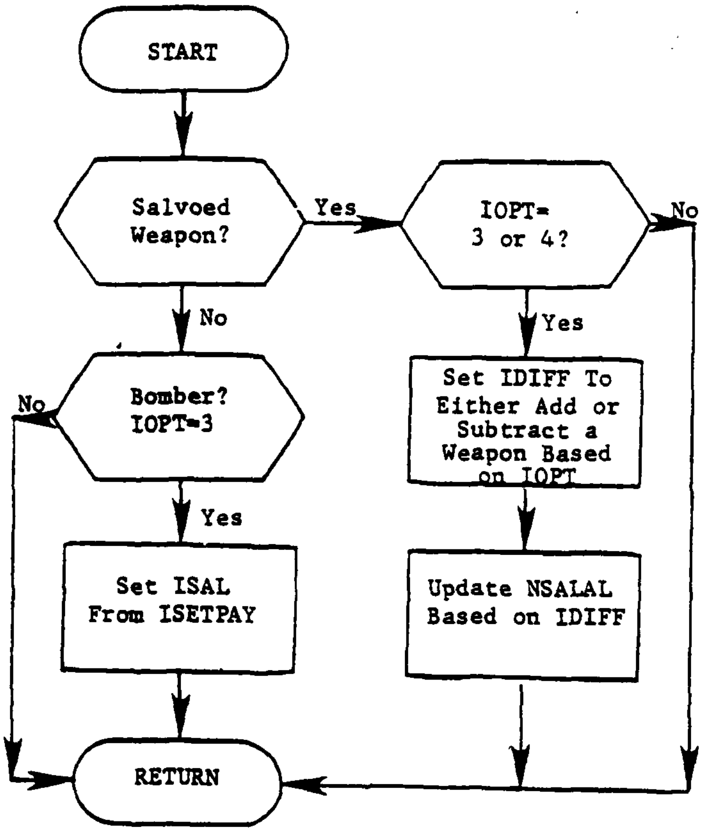


Figure 30. Subroutine ADDSAL

3.8.2 Subroutine ASGOUT

PURPOSE:	To update allocation assignment records in the integrated data base
ENTRY POINTS:	ASGOUT, ASGIN, ASGSAV
FORMAL PARAMETERS:	None
COMMON BLOCKS:	C10, C30, DYNAMI, PAYSAV, TGTSAV, WEPSAV, KASG, LGR
SUBROUTINES CALLED:	DIRECT, DLETE, MODFY, NEXTTT, STORE, HEAD, GLOG, SLOG
CALLED BY:	MULCON

Method:

First a logical switch is set for each new weapon assignment to indicate it is unassigned. Next each old assignment is compared to the new assignments to see if all values, except RVAL, of the old assignment are equal to a new assignment. If they are, the RVAL attribute is deleted. Finally an ASSIGN record is created for all the new assignments for which there is no match.

ASGOUT contains the internal array LGREF. This array is initialized to 0. Then, each time an assignment is made to a group, the reference code of the ASSIGN record is stored in LGREF. Prior, to the storing of any ASSIGN record, if LGREF is non-zero, the last assignment to the subject group which was stored is retrieved and then the new ASSIGN record reference code is stored in LGREF. LGREF is set to zero only if the reference code found in LGREF is the same as that for the record to be deleted.

Subroutine ASGOUT has 3 entry points:

- o ASGOUT: Writes assignment data to temporary file 25 after each pass during an assignment.
- o ASGIN: Reads assignment data from temporary file 25 before each pass during an assignment.
- o ASGSAV: When assignment is complete, writes assignment data to the IDS base.

Subroutine ASGOUT is illustrated in figure 31.

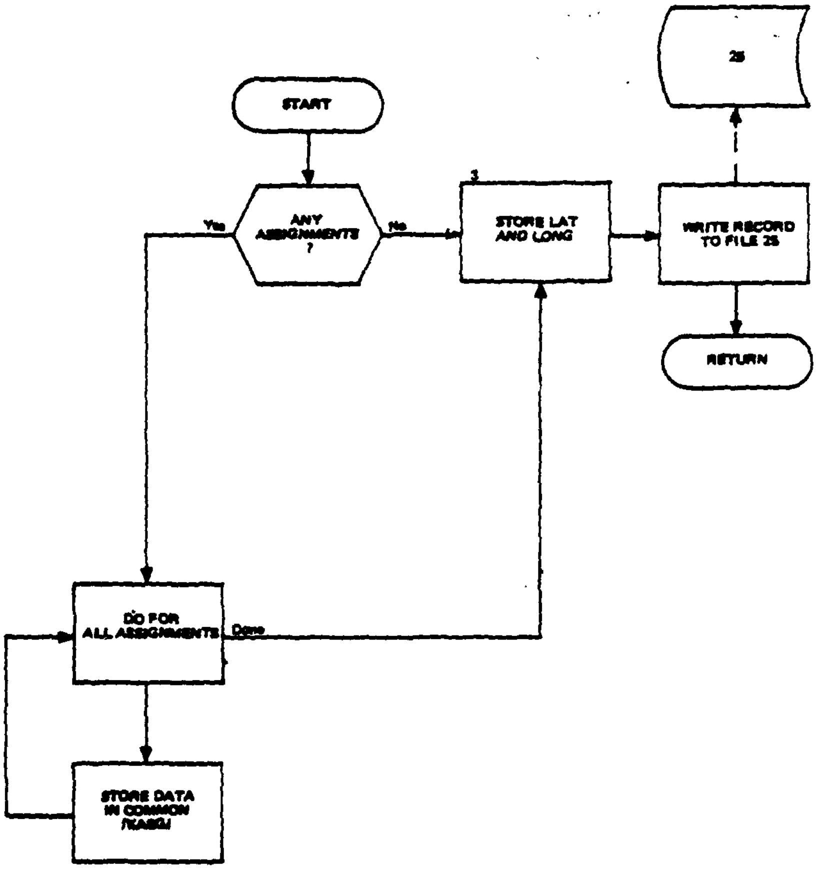


Figure 31. Subroutine ASGOUT

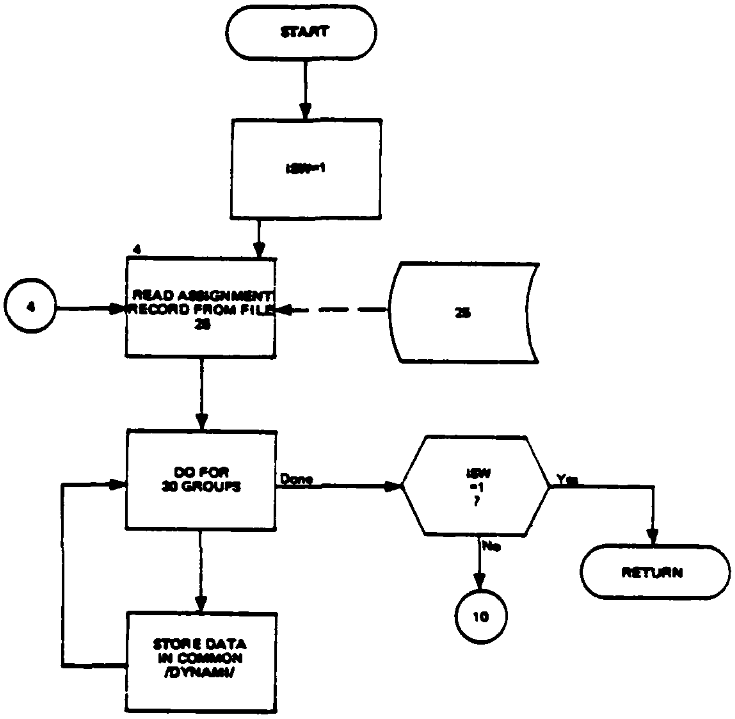

START
ISW=1
4
4
READ ASSIGNMENT RECORD FROM FILE 25
25
DO FOR 30 GROUPS
Done
ISW =1 ?
Yes
No
10
RETURN
STORE DATA IN COMMON /DYNAMI/

Figure 31.1. Subroutine ASGOUT (Entry ASGIN)

AD-A085 289 COMMAND AND CONTROL TECHNICAL CENTER WASHINGTON DC F/G 15/7
THE CCTC QUICK-REACTING GENERAL WAR GAMING SYSTEM (QUICK). PROG--ETC(U)
APR 80

UNCLASSIFIED NL

2 OF 3
AD A085289

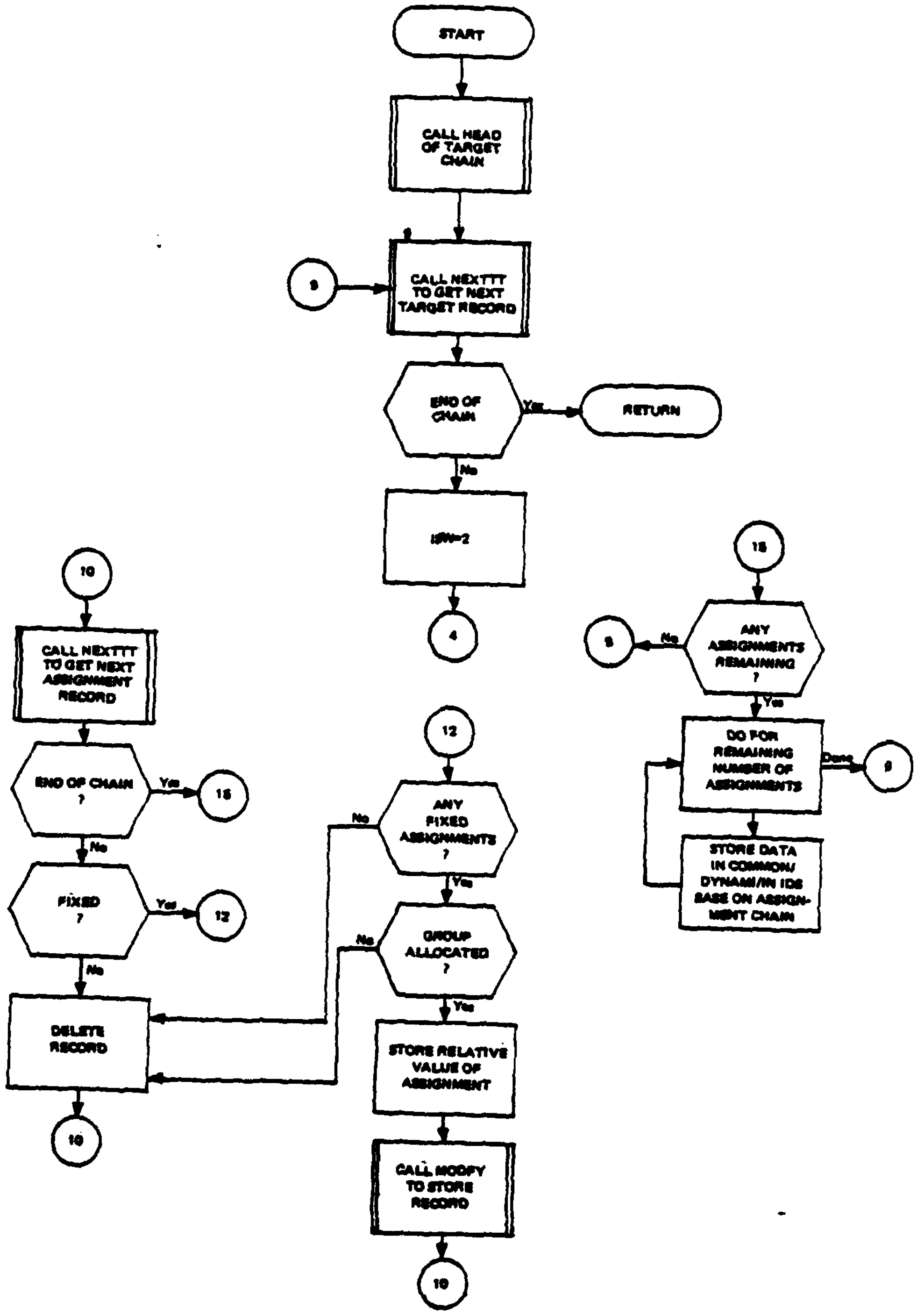

START
CALL HEAD OF TARGET CHAIN
9
CALL NEXTTT TO GET NEXT TARGET RECORD
END OF CHAIN
Yes
RETURN
No
ISW=2
4
10
CALL NEXTTT TO GET NEXT ASSIGNMENT RECORD
END OF CHAIN ?
Yes
15
No
FIXED ?
Yes
12
No
DELETE RECORD
10
12
ANY FIXED ASSIGNMENTS ?
No
Yes
GROUP ALLOCATED ?
No
Yes
STORE RELATIVE VALUE OF ASSIGNMENT
CALL MODFY TO STORE RECORD
10
15
ANY ASSIGNMENTS REMAINING ?
No
9
Yes
DO FOR REMAINING NUMBER OF ASSIGNMENTS
Done
9
STORE DATA IN COMMON/ DYNAMI/IN IDS BASE ON ASSIGNMENT CHAIN

Figure 31.2. Subroutine ASGOUT (Entry ASGSAV)

3.8.3 Subroutine BOMPRM

PURPOSE:	The purpose of this routine is to maintain the array containing the fraction of weapons all allocated from each group which are ASMs.
ENTRY POINT:	BOMPRM
FORMAL PARAMETERS:	IDIFF = -1 if weapons are being deleted +1 if weapons are being added
COMMON BLOCKS:	C33, DYNAMI, NALLY, PAYSAV, SALVO, WEPSAV
SUBROUTINES CALLED:	None
CALLED BY:	MULCON, SCNDGD

Method:

This routine merely updates the ASM fraction array FASM in common block /SALVO/. The important local variables are:

DONE = a logical array set true if a weapon has already been processed to update FASM

TOTW = total number of weapons allocated from a group on the target

TASM = number of ASMs from a group allocated on the target

FASM(G) is the fraction of currently allocated weapons from group G which are ASMs.

The factor FASM is updated whenever the state of the allocation changes. These changes occur when allocations from a previous pass are removed and when the allocation from the present pass is output. Thus, BOMPRM is called from subroutine MULCON on each target, and by subroutine SCNDGD for each target after the first pass.

Subroutine BOMPRM has one formal parameter IDIFF. If weapons are being removed (previous pass's allocation), then the value of IDIFF is -1. If weapons are being added, then IDIFF is equal to +1. In subroutine SCNDGD, the call to BOMPRM with IDIFF equal to -1 is made after reading the last pass allocation, just prior to the update of the running allocation sums. The call from MULCON with IDIFF equal to +1 is made just prior to the running sum update.

Upon entry to subroutine BOMPRM, the routine checks variable NBLN in /C33/. If this variable is negative, the allocation in /DYNAMI/ was made by subroutine DEFALOC and contains no bomber weapons. In this case, the subroutine returns with no further processing. If the allocation was made by subroutine STALL, the IG array of /DYNAMI/ is checked. For each allocation in this array, the variable KORRX of /DYNAMI/ is checked to determine if the weapon is a bomber. If not, the next entry in the IG array is checked. If the weapon is a bomber, then the value of GSEASM for the group is checked (in /WEPSAV/). If GSEASM is equal to zero or one, then FASM is set to GSEASM and processing continues with the next entry in the IG array. Otherwise, FASM is updated for the group.

The ISAL array of /DYNAMI/ contains the indicator of bomb or ASM allocation (for bomber groups only. This array is defined differently for missile groups). If the value is zero, a gravity bomb was allocated. A value of one signifies the use of an ASM.

The total number of weapons from the group which are currently allocated is kept in array RNALL of common /NALLY/. This array is updated twice for each target, just following the call on BOMPRM.

Using these variables, the value of FASM is updated as follows:

Define: TASM = number of ASMs allocated from group G (as determined from the ISAL array) on current target

TOTW = number of weapons allocated from group G on current target

Then,

$$\text{FASM}_{\text{new}} = \frac{(\text{FASM}_{\text{old}} * \text{RNALL}) + \text{TASM}}{\text{RNALL} + \text{TOTW}}$$

Note that the variables FASM and RNALL in the above equation are arrays indexed by the group number G.

Figure 32 displays the logic of subroutine BOMPRM.

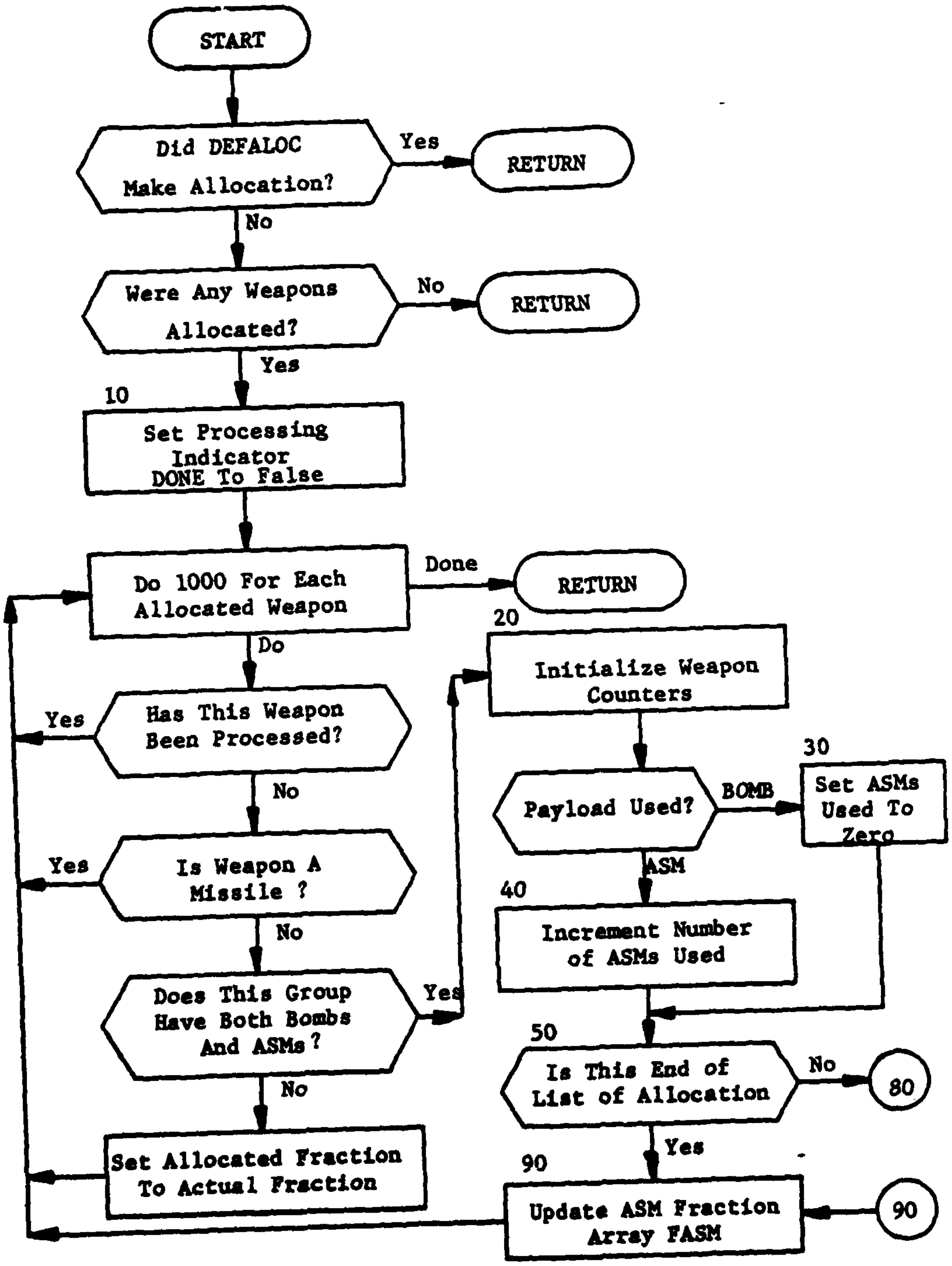

START
Did DEFALOC Make Allocation?
Yes
RETURN
No
Were Any Weapons Allocated?
No
RETURN
Yes
10
Set Processing Indicator DONE To False
Do 1000 For Each Allocated Weapon
Done
RETURN
Do
Has This Weapon Been Processed?
Yes
No
Is Weapon A Missile ?
Yes
No
Does This Group Have Both Bombs And ASMs ?
Yes
No
Set Allocated Fraction To Actual Fraction
20
Initialize Weapon Counters
Payload Used?
BOMB
30
Set ASMs Used To Zero
ASM
40
Increment Number of ASMs Used
50
Is This End of List of Allocation
No
80
Yes
90
Update ASM Fraction Array FASM
90

Figure 32. Subroutine BOMPRM (Part 1 of 2)

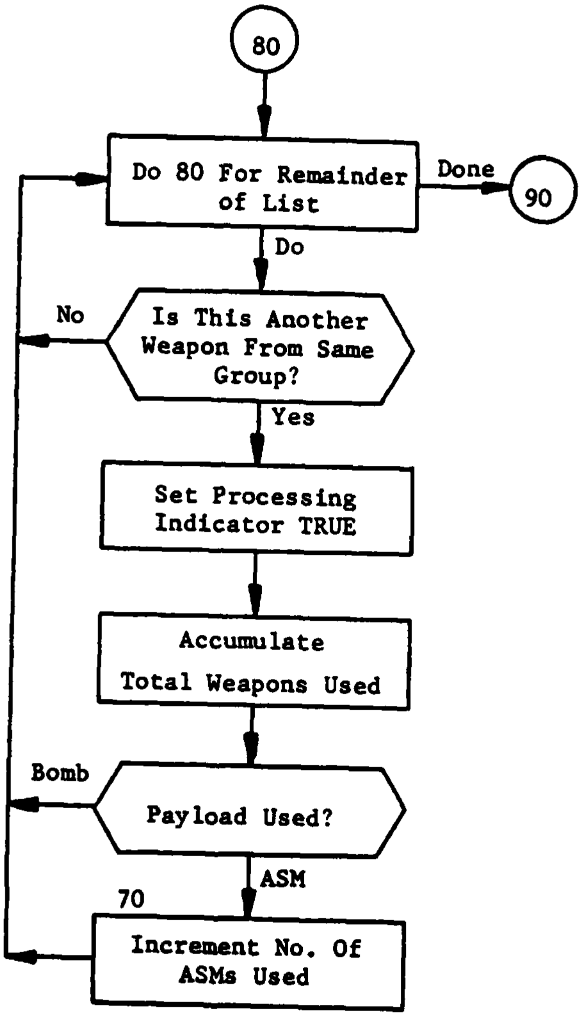

80
Do 80 For Remainder of List
Done
90
Do
No
Is This Another Weapon From Same Group?
Yes
Set Processing Indicator TRUE
Accumulate Total Weapons Used
Bomb
Payload Used?
ASM
70
Increment No. Of ASMs Used

Figure 32. (Part 2 of 2)

CONTENTS OF THIS PAGE INTENTIONALLY DELETED

3.8.5 Subroutine PRNTALL

PURPOSE:	This routine provides a way of calling the print subroutine PRNTNOW that is conditional on the print control flags set by PRNTCON.
ENTRY POINTS:	PRNTALL
FORMAL PARAMETERS:	IOPT - Print option number
COMMON BLOCKS:	C30, CONTRO, PRNTCN
SUBROUTINES CALLED:	PRNTNOW, TIMEME
CALLED BY:	MULCON, WAD, WADOUT, FRSTGD, RESVAL, DEFALOC, SETPAY

Method:

To provide convenient control over prints in program ALOC almost all print statements are contained in subroutine PRNTNOW. They are activated by calling PRNTNOW(IOPT) for the appropriate print option IOPT. If it is desired to place the print under data-input control so that the print will not appear unless a specific print request is included in the data deck, this can be accomplished by calling PRNTNOW via a call on PRNTALL. PRNTALL executes the request on PRNTNOW only if the print control subroutine PRNTCON has set the corresponding print control flag IDO(IOPT) active (i.e., = 3).

For each call PRNTALL first checks to see if the print has been set active by PRNTCON. If not, it immediately RETURNs (statement 2) to minimize time wasted on inoperative print calls.

If the particular print is active, PRNTALL immediately calls TIMEME (statement 1) to stop the clock which records active time in the program. This makes it possible to do a test run with an unusual number of prints and still obtain a valid estimate of what the running time would be without such prints. After the call on PRNTNOW, PRNTALL reactivates the clock before returning to the main program.

Before each print option (except 26), PRNTALL prints a heading identifying the optional print.

Subroutine PRNTALL is illustrated in figure 34.

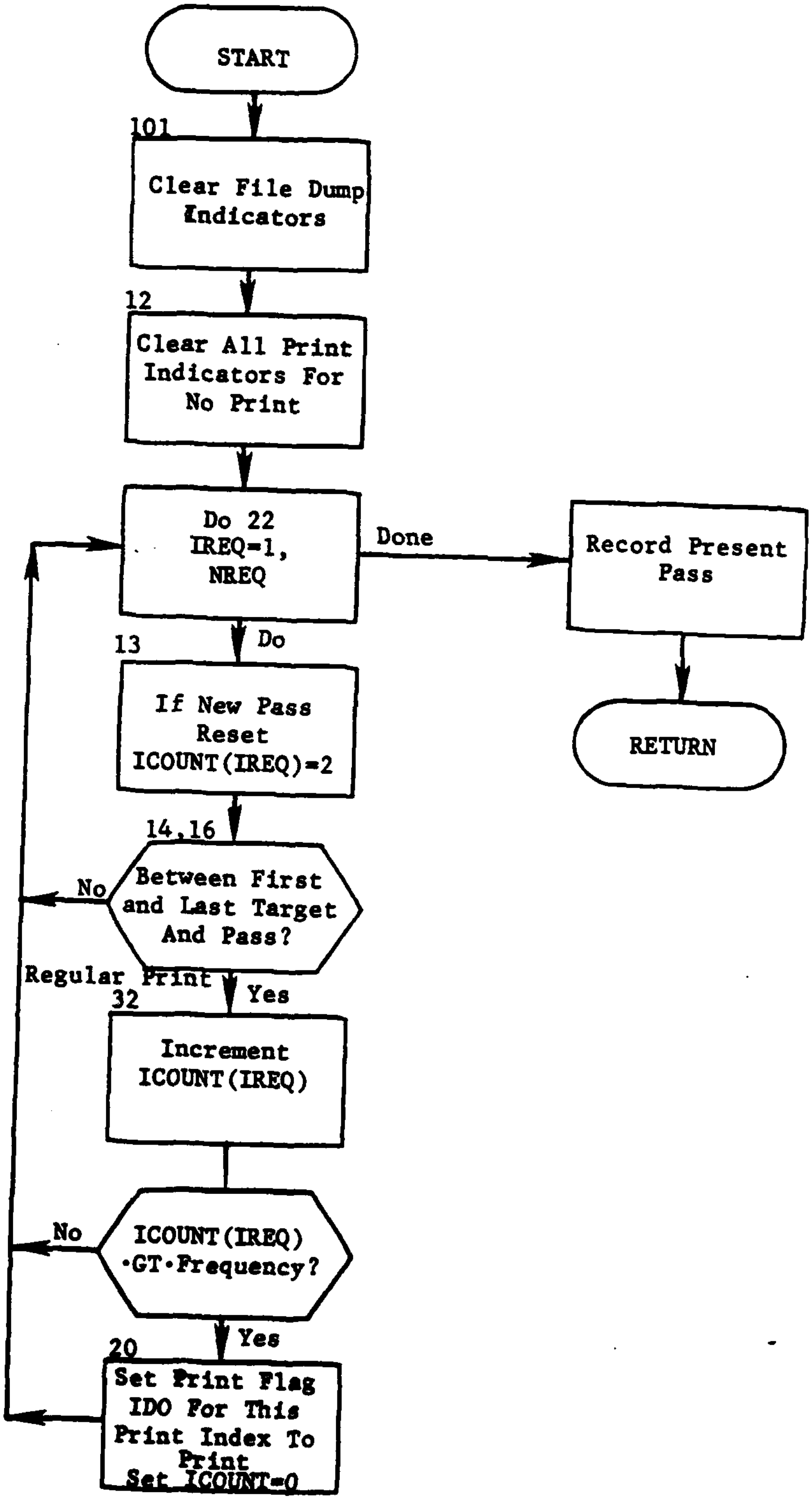

START
101
Clear File Dump Indicators
12
Clear All Print Indicators For No Print
Do 22 IREQ=1, NREQ
Done
Record Present Pass
RETURN
Do
13
If New Pass Reset ICOUNT(IREQ)=2
14,16
Between First and Last Target And Pass?
No
Regular Print
Yes
32
Increment ICOUNT(IREQ)
ICOUNT(IREQ) ·GT·Frequency?
No
Yes
20
Set Print Flag IDO For This Print Index To Print
Set ICOUNT=0

Figure 35. Subroutine PRNTCON

3.8.7 Subroutine PRNTNOW

PURPOSE:	To produce optional printed output.
ENTRY POINTS:	PRNTNOW
FORMAL PARAMETERS:	IOPT - Print option number
COMMON BLOCKS:	ALERUN, C30, C33, DYNAMI, NALLY, DAYSAV, PREMS, PRTMUL, SALVO, SURPW, TGTSAV, LACB, PAYOFF, WADFIN, WADOTX, WADWPN, WEPSAV, WPFIX, WTS, NOWPS,
SUBROUTINES CALLED:	ABORT, PRNTOD, PRNTOF, PRNTOS, TIMEME
CALLED BY:	PRNTALL, MULCON

Method:

The formal parameter IOPT determines which print is produced. The result of the alternatives appear in the Users Manual, UM 9-77, Volume III. Options 1, 12, 13, 26, 27, and 28 require a subroutine be called which contains the print function.

Subroutine PRNTNOW is illustrated in figure 36.

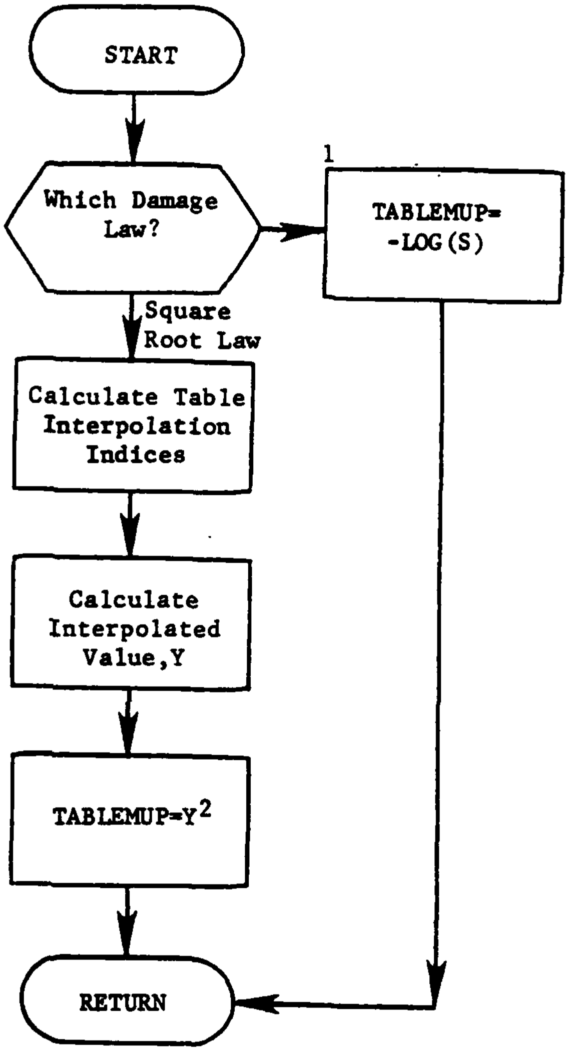


Figure 37. Function TABLEMUP

3.9 Subroutine FRSTGD[*]

PURPOSE: Assemble allocation data on the first pass.

ENTRY POINTS: FRSTGD

FORMAL PARAMETERS: None

COMMON BLOCKS: C10, C15, C30, C33, DYNAMI, FIRST, GRPHDR, GRPSTF, INITSW, PAYSAV, PNAV, SALVO, TARREF, TGTSAV, WADWPN, WEPSAV, WPFIX, XFPX, QTKCOM, IRTLEN

SUBROUTINES CALLED: CRDCAL, DIRECT, FLGCHK, HDFND, HEAD, INICRD, MODFY, NEXTTT, PKCALC, PRNTAL, RECON, RETRV, TGTCRD, TIMEME, TGTIN

CALLED BY: MULCON

Method:

This routine processes each target in target number order. Each call causes the next target to be retrieved. Since each record on the target list points to either a target or a complex record, the next step in the process is to retrieve the remainder of the target data. Next the weapon data is acquired. This process depends to a great extent upon whether the user has saved file 15 from a previous run of ALOC. If so, this file is read in and unpacked. If not, the file is created by cycling through the weapon groups and calculating the various needed quantities. Much of the group data needed for this process is contained on file 23 where it was stored by DATGRP. If the user has specified range modifications, any information which differs from that on file 15 is written on file 22 in the same format.

During this process, the INACTIVE array is set. This array has an entry for each group and is either set to 0 or 100. 0 implies that the group is available for allocation to the target. 100 indicates that the group is unavailable for one of several reasons: target out of range, time decay requirements, and flag location and MIRV restriction. This array is written onto unit 21.

The final step is to read in any fixed assignments to the target and update the assignment records.

Subroutine FRSTGD is illustrated in figure 38.

[*] First subroutine of segment FGD.

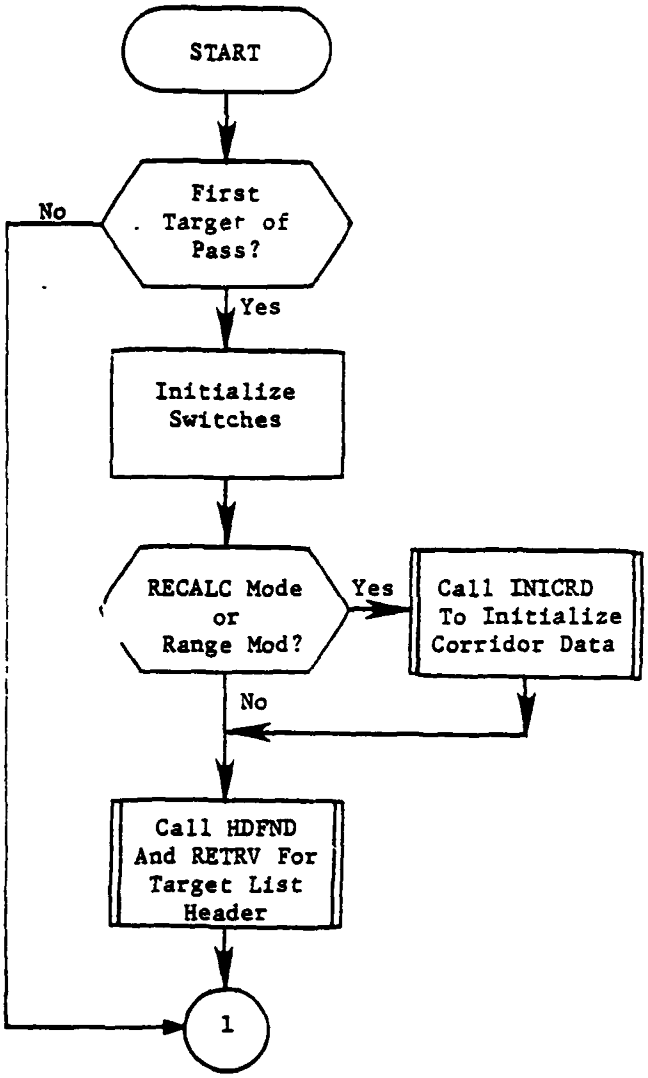

START
First Target of Pass?
No
Yes
Initialize Switches
RECALC Mode or Range Mod?
Yes
Call INICRD To Initialize Corridor Data
No
Call HDFND And RETRV For Target List Header
1

Figure 38. Subroutine FRSTGD (Part 1 of 11)

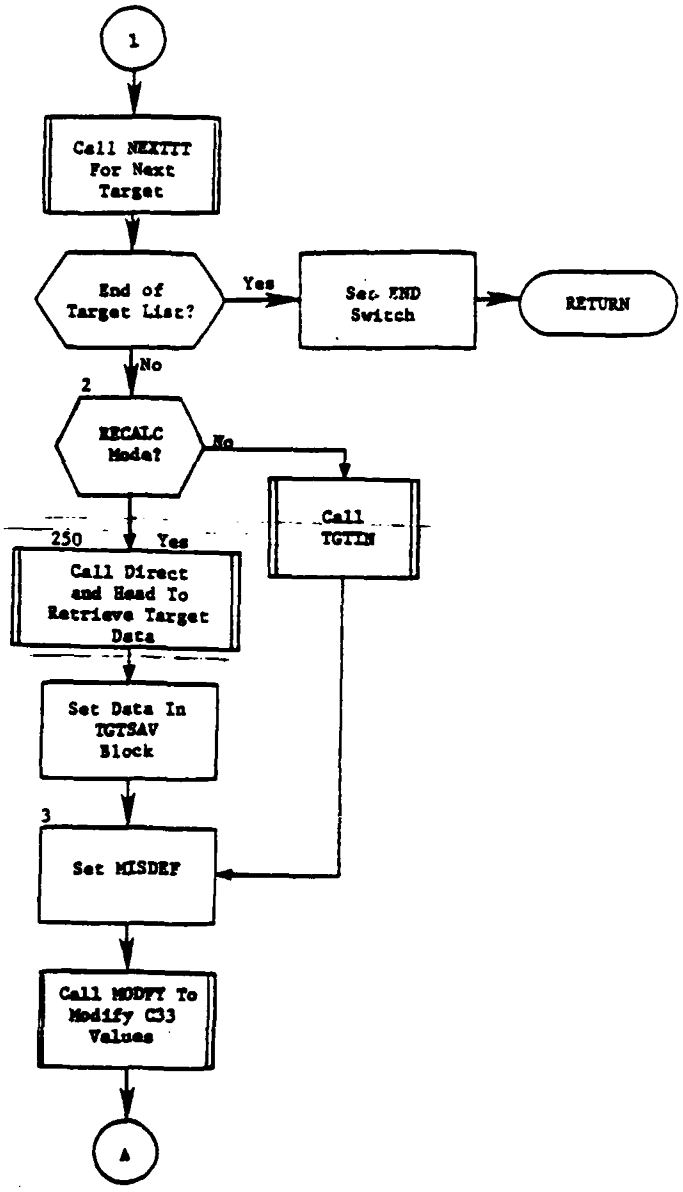


Figure 38. (Part 2 of 11)

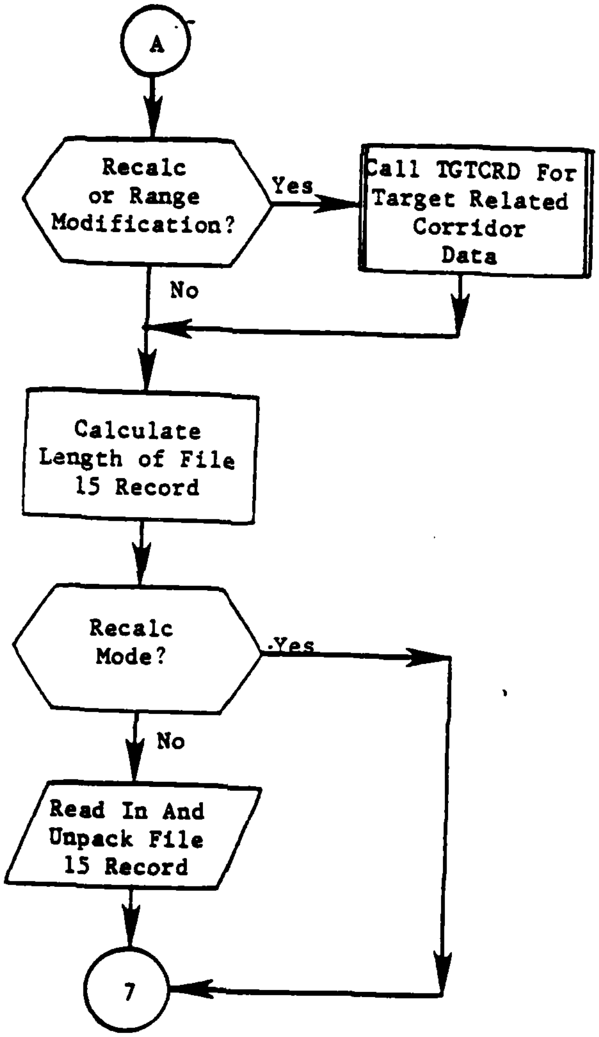

A
Recalc or Range Modification?
Yes
Call TGTCRD For Target Related Corridor Data
No
Calculate Length of File 15 Record
Recalc Mode?
Yes
No
Read In And Unpack File 15 Record
7

Figure 38. (Part 3 of 11)

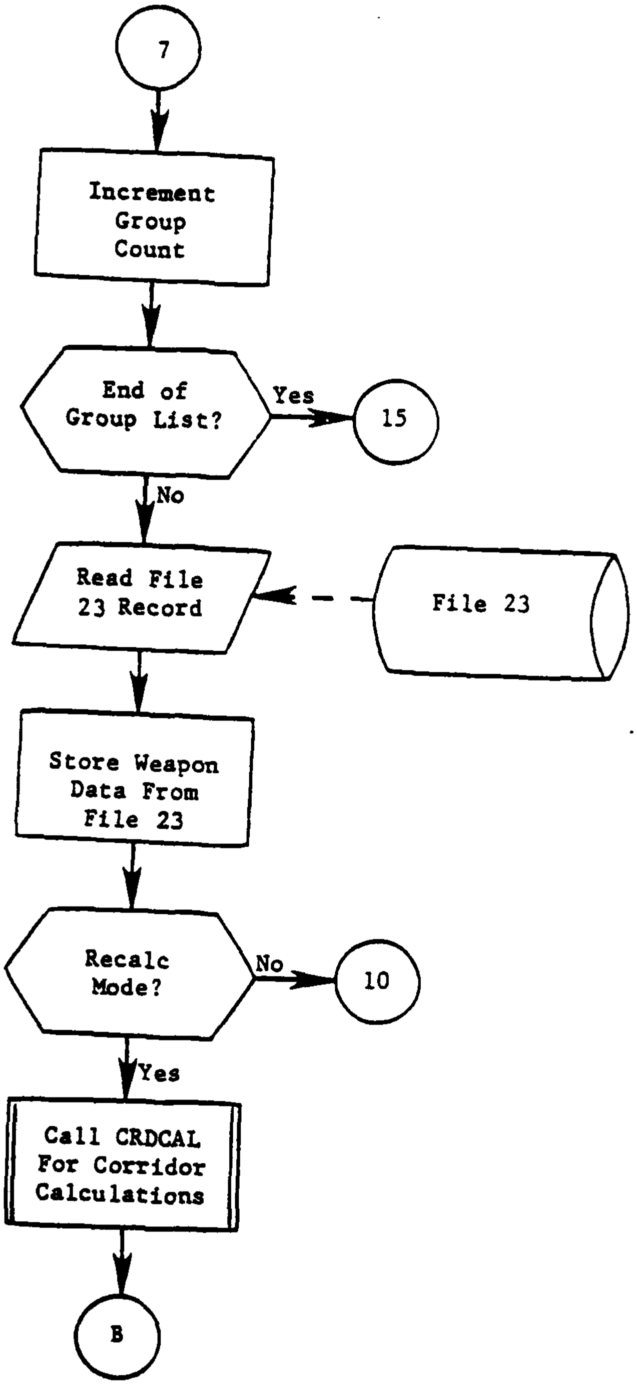


Figure 38. (Part 4 of 11)

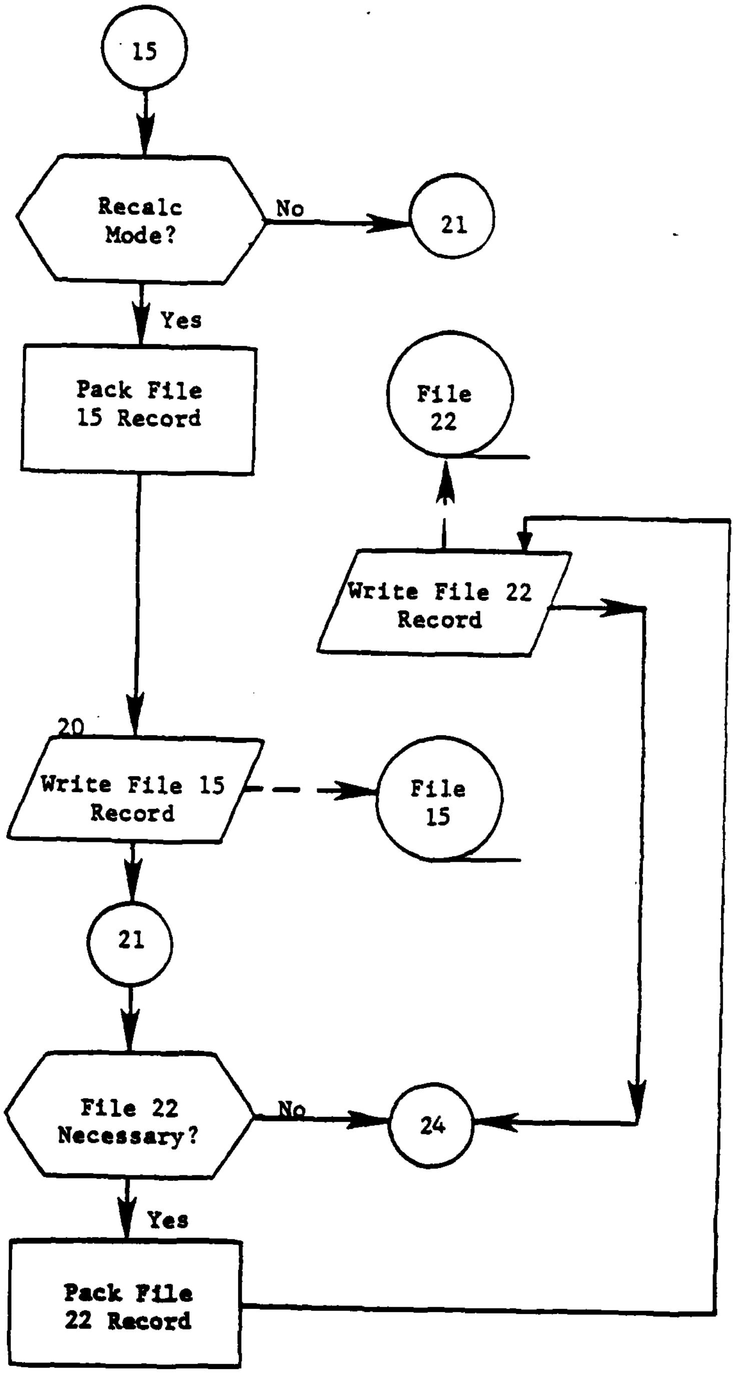


Figure 38. (Part 9 of 11)

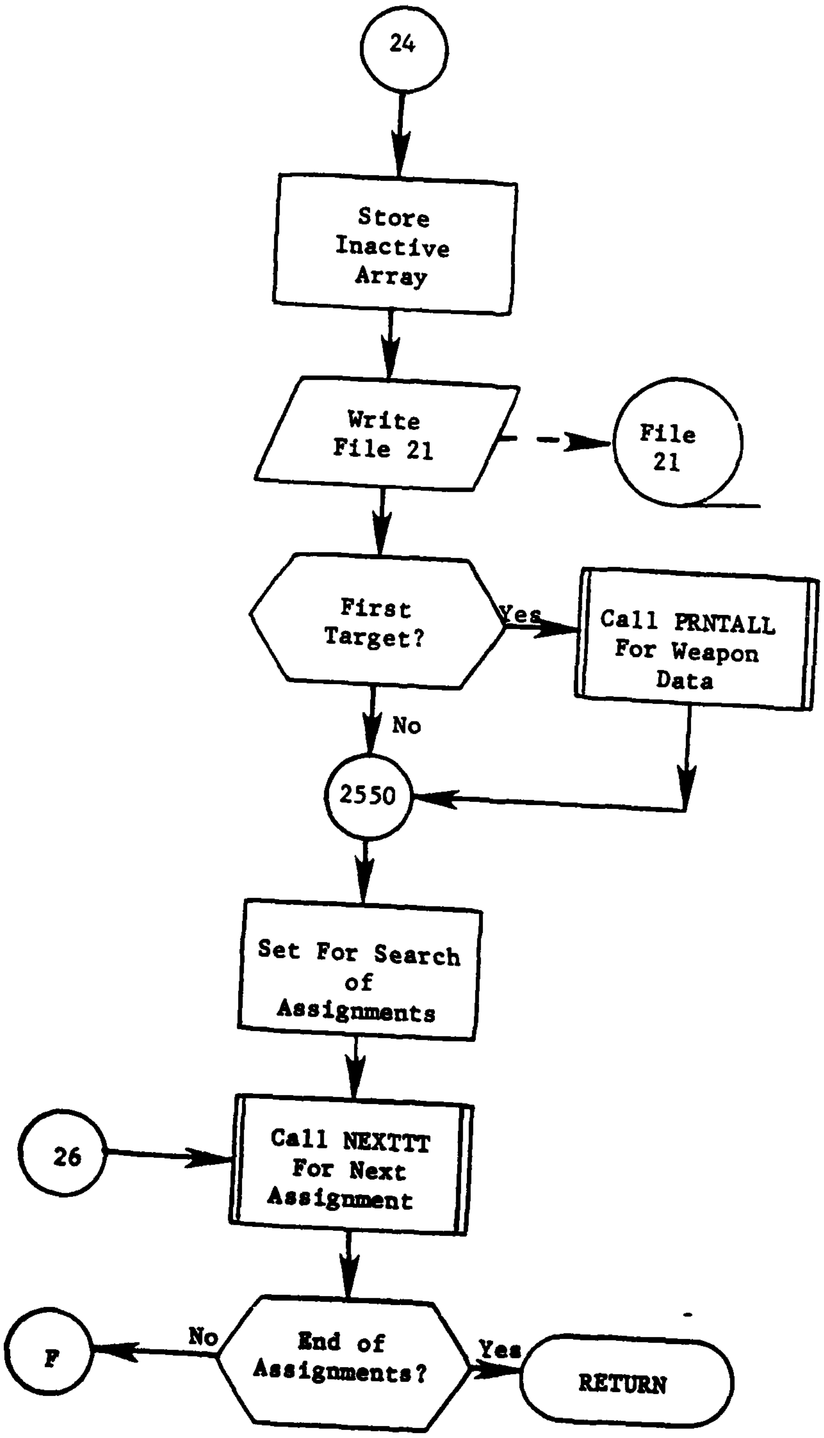


Figure 38. (Part 10 of 11)

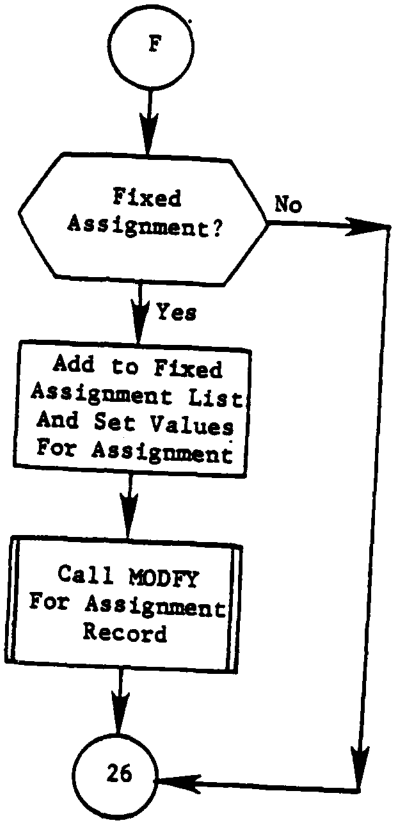

F
Fixed Assignment?
No
Yes
Add to Fixed Assignment List And Set Values For Assignment
Call MODFY For Assignment Record
26

Figure 38. (Part 11 of 11)

3.9.1 Subroutine CRDCAL

PURPOSE:	Calculate penetration probability and pick bomber corridor
ENTRY POINTS:	CRDCAL
FORMAL PARAMETERS:	JORR - Corridor picked XEN - Penetration probability ZOA -- Time of arrival ICLSS - Class index of group
COMMON BLOCKS:	C30, CORSTF, PAYSAV, REFPNT, TGTSAV, WADWPN, WEPSAV, XFPX
SUBROUTINES CALLED:	DISTF, EXP, TOFM
CALLED BY:	FRSTGD

Method:

First a check is made for match-ups of naval weapons to naval targets. Next, if the group is a missile, its corridor is set to zero and its penetration probability and time of arrival are calculated.

For bomber groups, first the PKNAV and IPENMO are checked to see if group is restricted to corridors 2 and 1 respectively. Otherwise, each corridor in turn is examined to see if it would provide the best route to the target. Corridors which would make the distance too great are rejected. For the remainder, the penetration probability is calculated with the bomber using low altitude flight as much as possible. The highest penetration corridor is chosen.

Subroutine CRDCAL is illustrated in figure 39.

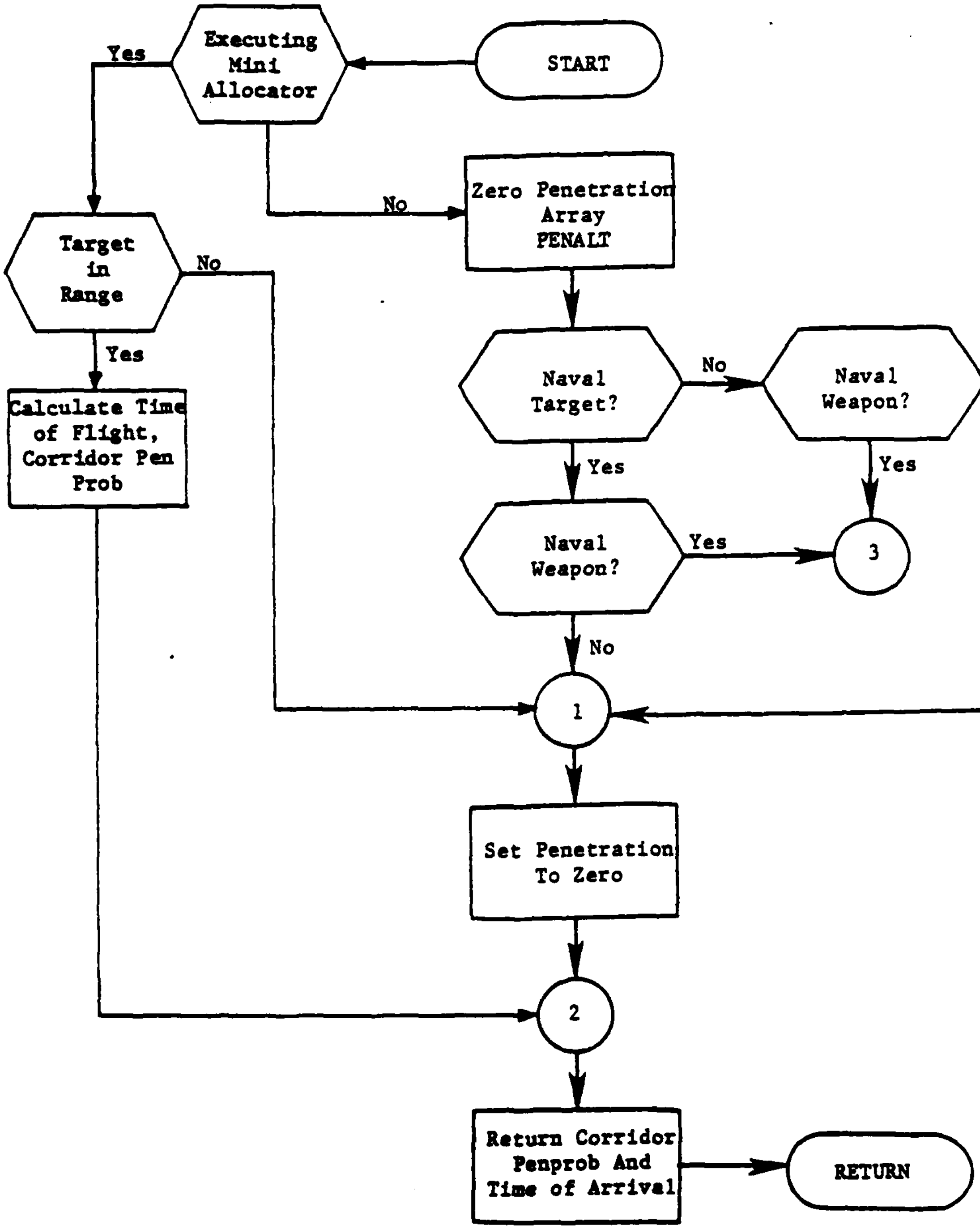


Figure 39. Subroutine CRDCAL (Part 1 of 6)

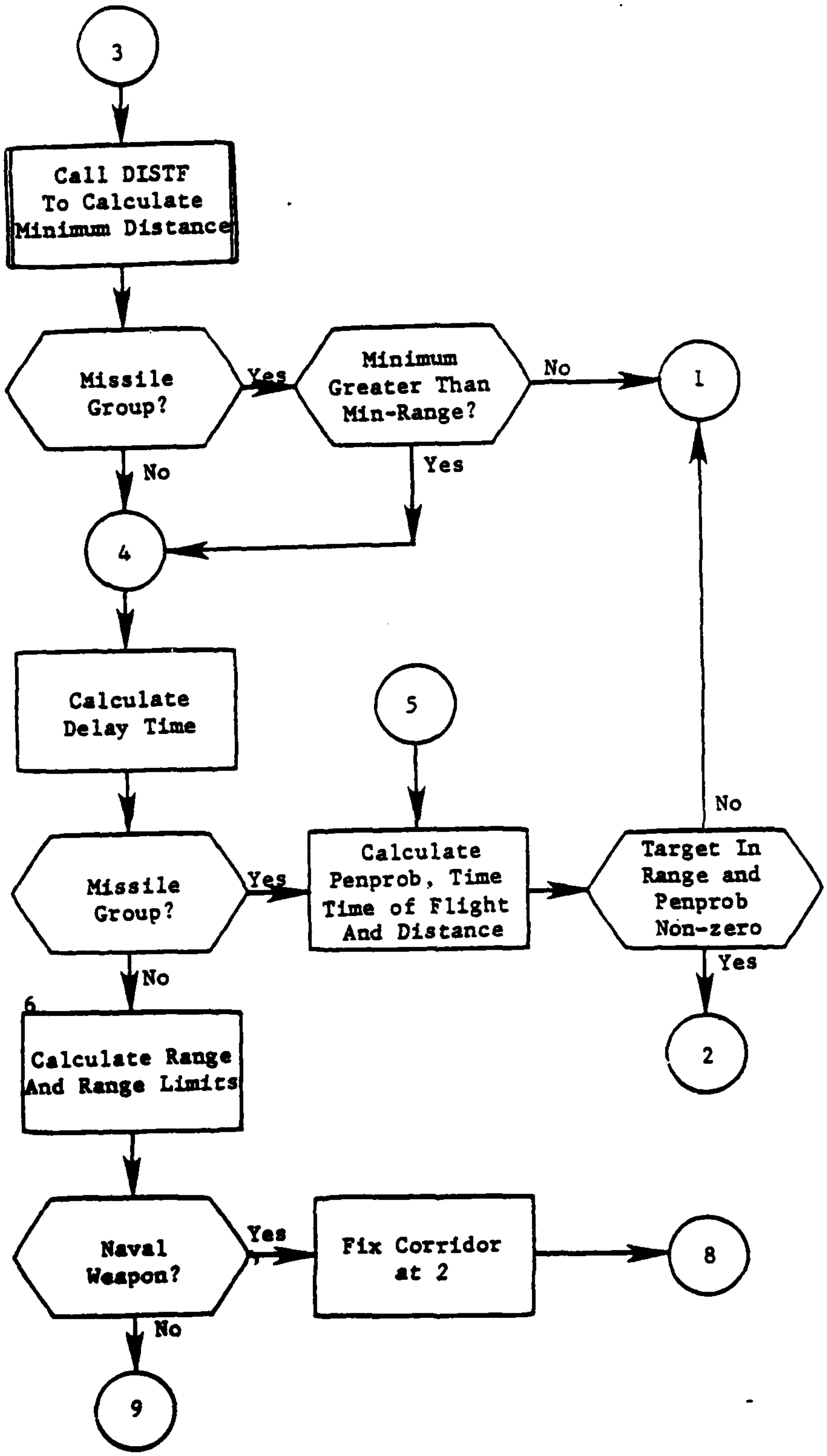

3
Call DISTF To Calculate Minimum Distance
Missile Group?
Yes
Minimum Greater Than Min-Range?
No
1
No
Yes
4
Calculate Delay Time
5
Missile Group?
Yes
Calculate Penprob, Time Time of Flight And Distance
Target In Range and Penprob Non-zero
No
Yes
2
No
6
Calculate Range And Range Limits
Naval Weapon?
Yes
Fix Corridor at 2
8
No
9

Figure 39. (Part 2 of 6)

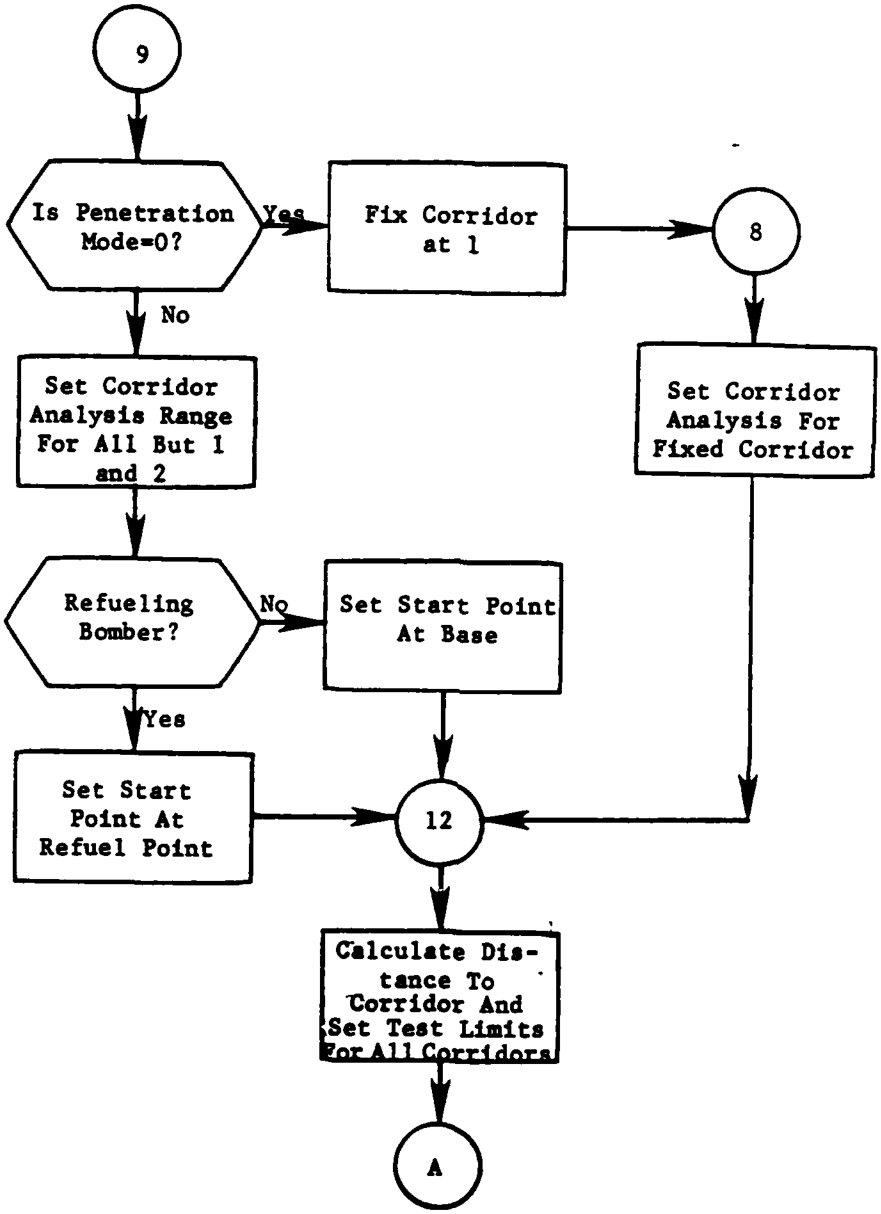

9
Is Penetration Mode=0?
Yes
Fix Corridor at 1
8
No
Set Corridor Analysis Range For All But 1 and 2
Set Corridor Analysis For Fixed Corridor
Refueling Bomber?
No
Set Start Point At Base
Yes
Set Start Point At Refuel Point
12
Calculate Distance To Corridor And Set Test Limits For All Corridors
A

Figure 39. (Part 3 of 6)

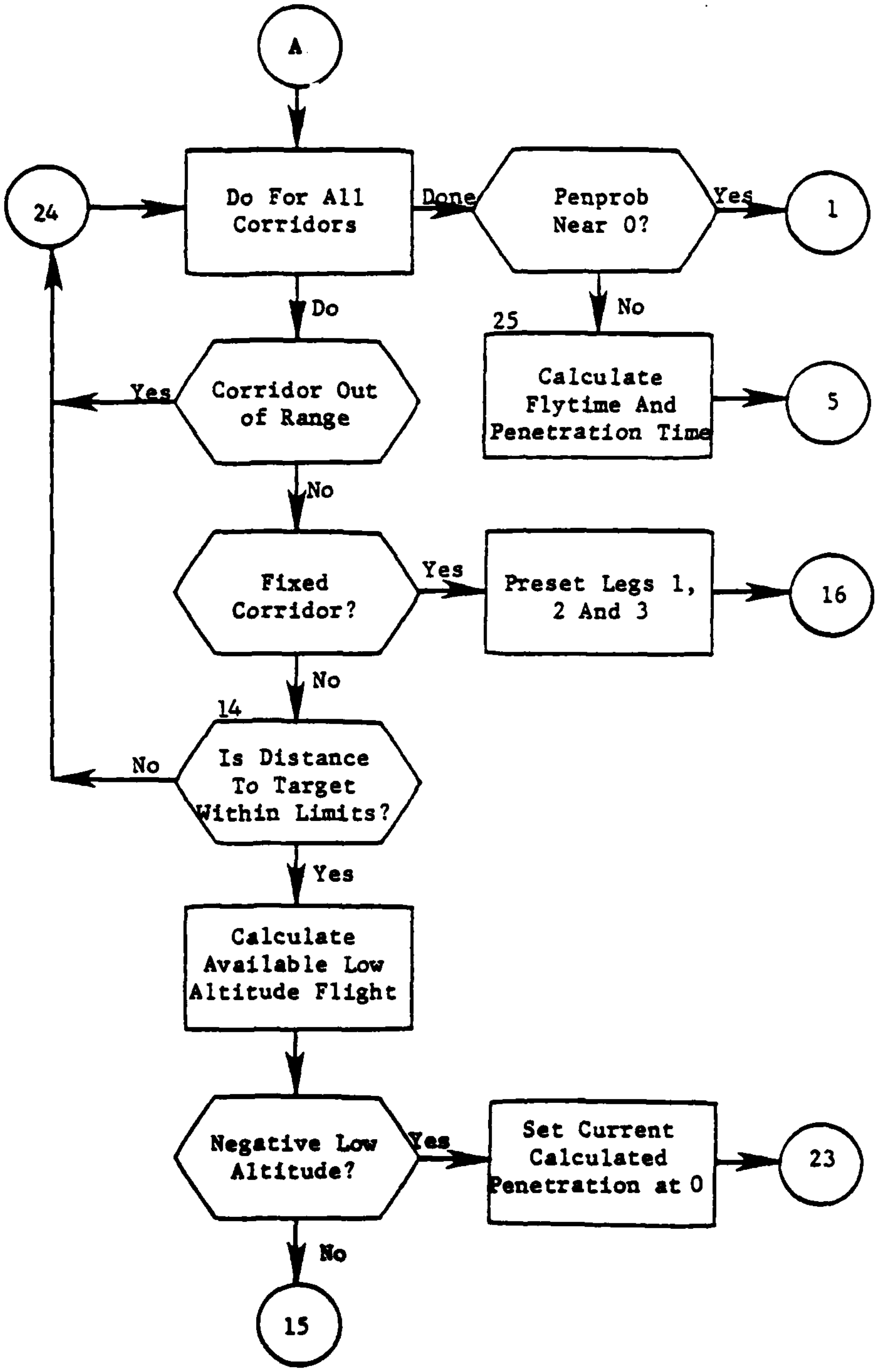


Figure 39. (Part 4 of 6)

3.9.2 Subroutine FLGCHK

PURPOSE:	To check flag, location and MIRV restrictions.
ENTRY POINTS:	FLGCHK
FORMAL PARAMETERS:	FIND - True if group not restricted False if group restricted
COMMON BLOCKS:	C30, CNCLS, GRPSTF, INITSW, TGTSAV
SUBROUTINES CALLED:	None
CALLED BY:	FRSTGD

Method:

This routine uses the logical arrays stored on file 25 by DATGRP. Each type of restriction is checked for correlation between the group and the current target.

Subroutine FLGCHK is illustrated by figure 40.

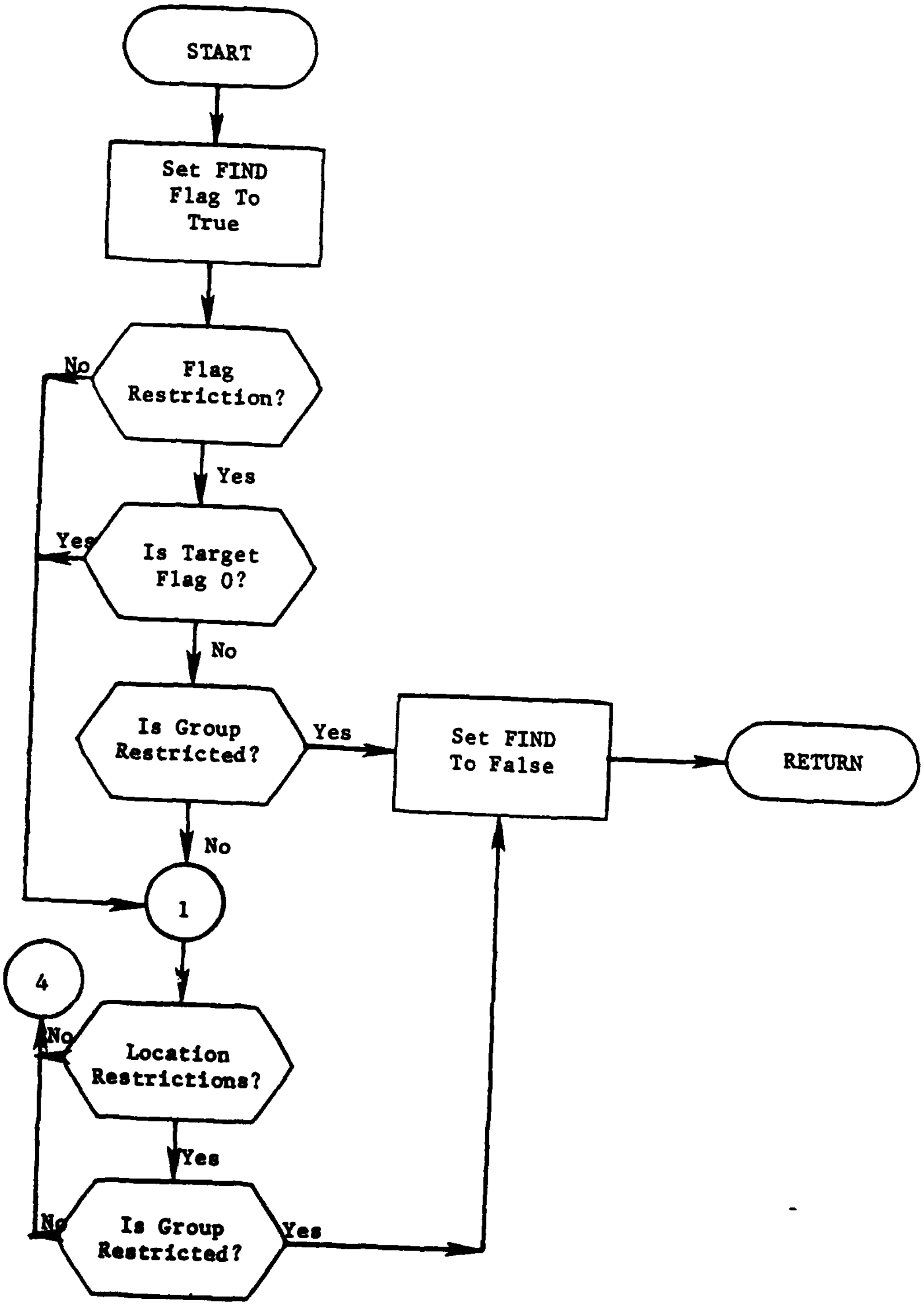


Figure 40. Subroutine FLGCHK (Part 1 of 3)

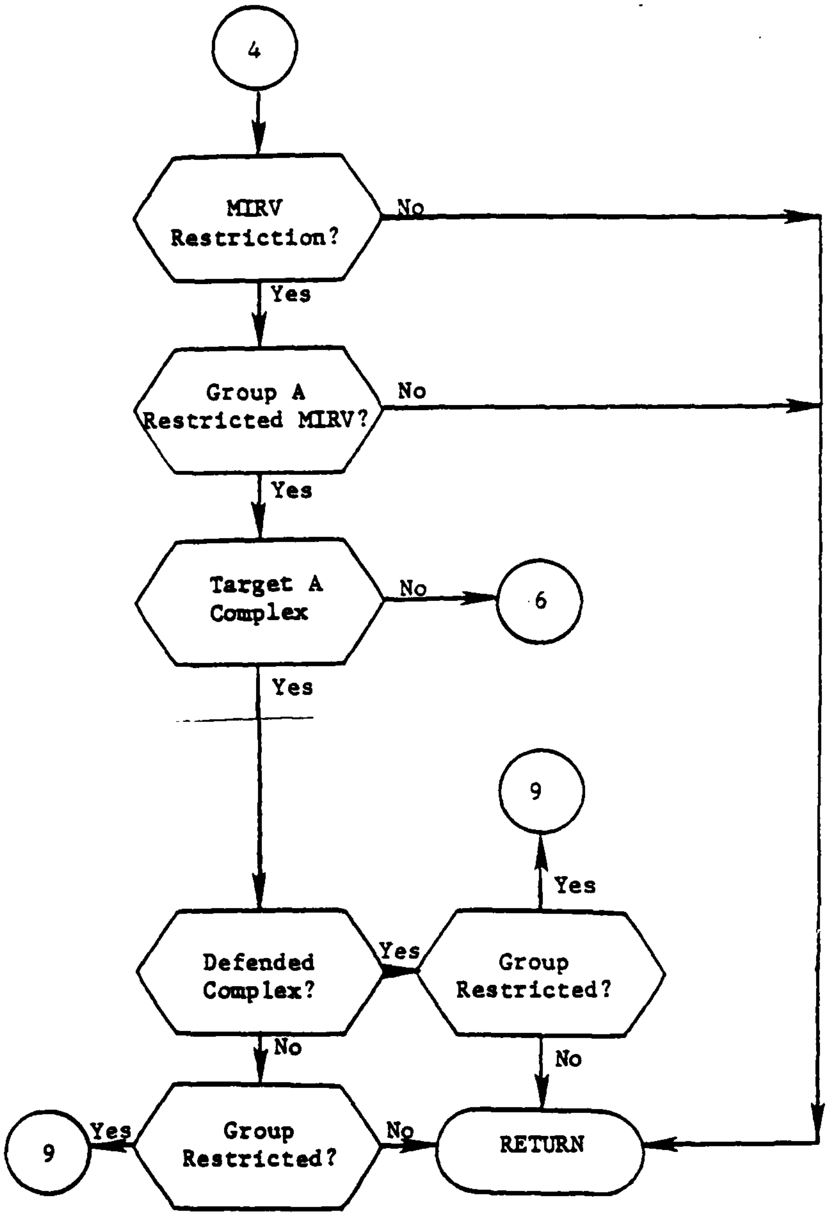


Figure 40. (Part 2 of 3)

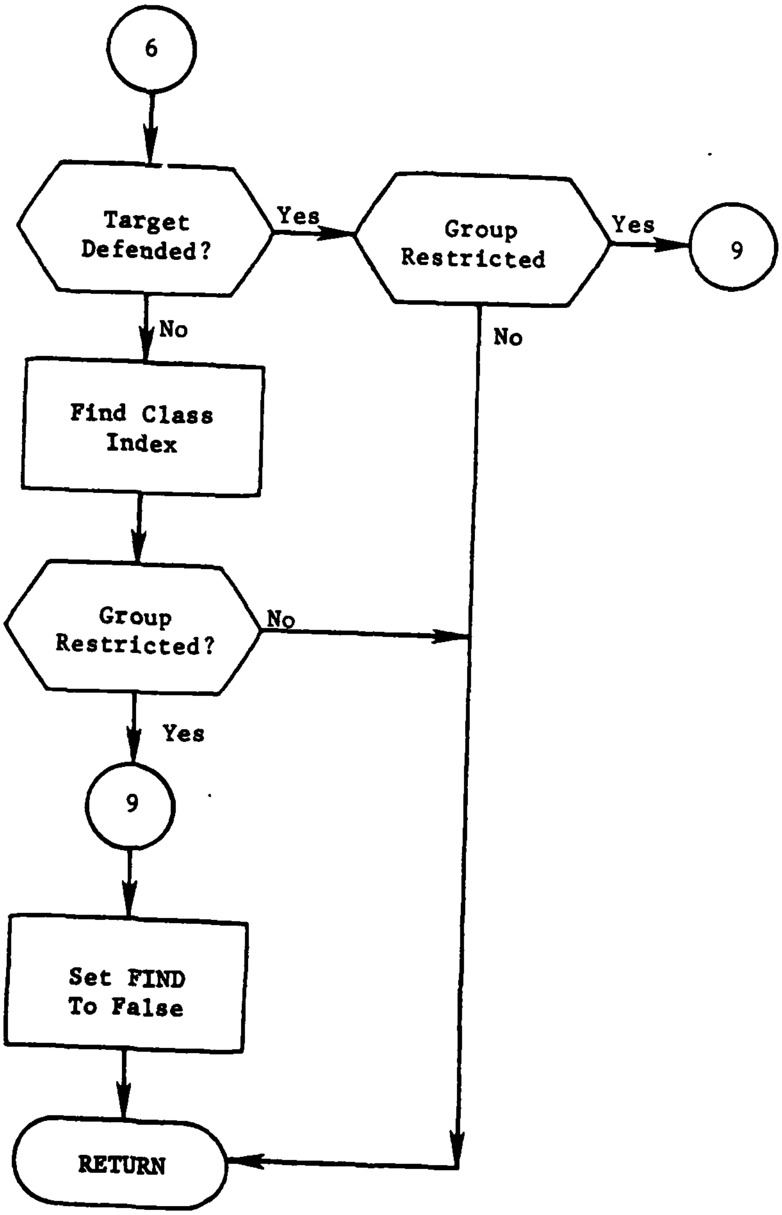


Figure 40. (Part 3 of 3)

CONTENTS OF PAGES 227 THROUGH 231 INTENTIONALLY DELETED

3.9.5 Subroutine PKCALC

PURPOSE: To calculate kill probabilities

ENTRY POINTS: PKCALC

FORMAL PARAMETERS:
STA - Array for normal PK
STB - Array for second PK
ICLSS - Class index of weapon group

COMMON BLOCKS: C30, PAYSAV, SALVO, TGTSAV, WEPSAV

SUBROUTINES CALLED: None

CALLED BY: FRSTGD

Method:

This routine calculates the kill probabilities (PK) of the weapon group against the target using a standard approximation to the circular coverage function. Two PK's are calculated for each hardness component of the target. The first PK is based upon the yield of the group as input and the group's CEP. The second PK varies according to the class of the weapon. For missiles it is the first PK adjusted by the number of warheads which make up the group yield (for any types but MRVs the first and second PK's are equal). For bombers the second PK reflects the PK of any ASMs onboard. Naval weapon's have their PK's input in the attribute GPKNAV. Average Destruction (AVDE) is calculated for bombers.

Subroutine PKCALC is illustrated in figure 43.

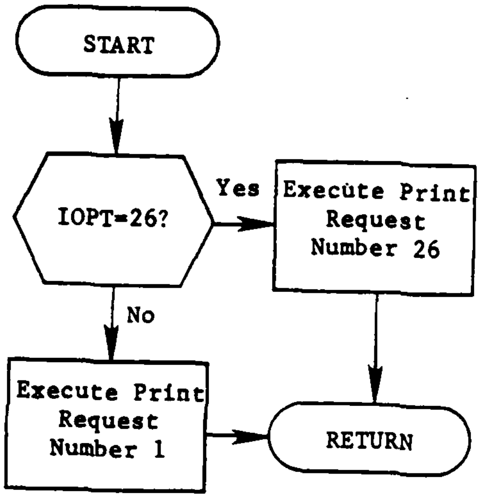

START
IOPT=26?
Yes
Execute Print Request Number 26
No
Execute Print Request Number 1
RETURN

Figure 44. Subroutine PRNTOF

3.9.7 Subroutine RECON

PURPOSE: To reconstruct data for the WADWPN block.

ENTRY POINTS: RECON

FORMAL PARAMETERS: ICLSS - class index of group, NOVAL - True if group is to be inactivated. False if group is to be activated

COMMON BLOCKS: C30, C45, PAYSAV, SALVO, SMATAD, TGTSAV, WADWPN, WEPSAV

SUBROUTINES CALLED: ALOG, SETPAY, TABLEMUP, VALTAR

CALLED BY: FRSTGD, SCNDGD

Method:

Subroutine SETPAY is called to select use of gravity bombs or ASMs from the bomber groups. If ASMs are selected probability STK 2 is used in the calculation of the survival probability STK and the MUP array. The MUP XMUP, SSIG, and RISK arrays are constructed according to the required formulae.

The SMAT array is loaded depending upon whether the group is a MIRV.

Subroutine RECON is illustrated in figure 45.

3.9.7.1 Subroutine RTAPCK

PURPOSE:	This routine handles the packing and unpacking of file 15 (22 if RANGMOD etc.). In addition, it handles the target data I/O on file 15.
ENTRY POINTS:	RTAPCK, RTAPIN, TGTIN, TGTOUT
FORMAL PARAMETERS:	INFL: WORD TO BE PACKED OR UNPACKED JDEF - ≥ 0 = target defended ≤ 0 = target undefended IO - 1 = unpacks 2 = packs
COMMON BLOCKS:	C30, IRTLEN, WADWPN, WEPSAV, PAYSAV, WPFIX, TGTSAV, QTKCOM
SUBROUTINES CALLED:	None
CALLED BY:	SCNDGD, FRSTGD

Method:

RTAPCK has 4 entry points:

- RTAPCK - Either packs (IO=2) or unpacks (IO=1) the contents of INFL using common blocks /WADWPN/ and /QTKCOM/ as the source or destination, respectively, of the data.
- RTAPIN - Calculates the entries for common block /IRTLEN/
- TGTLIN - Reads in target information from file 15.
- TGTOUT - Writes out target information to file 15.

Subroutine RTAPCK is illustrated in figure 45.1.

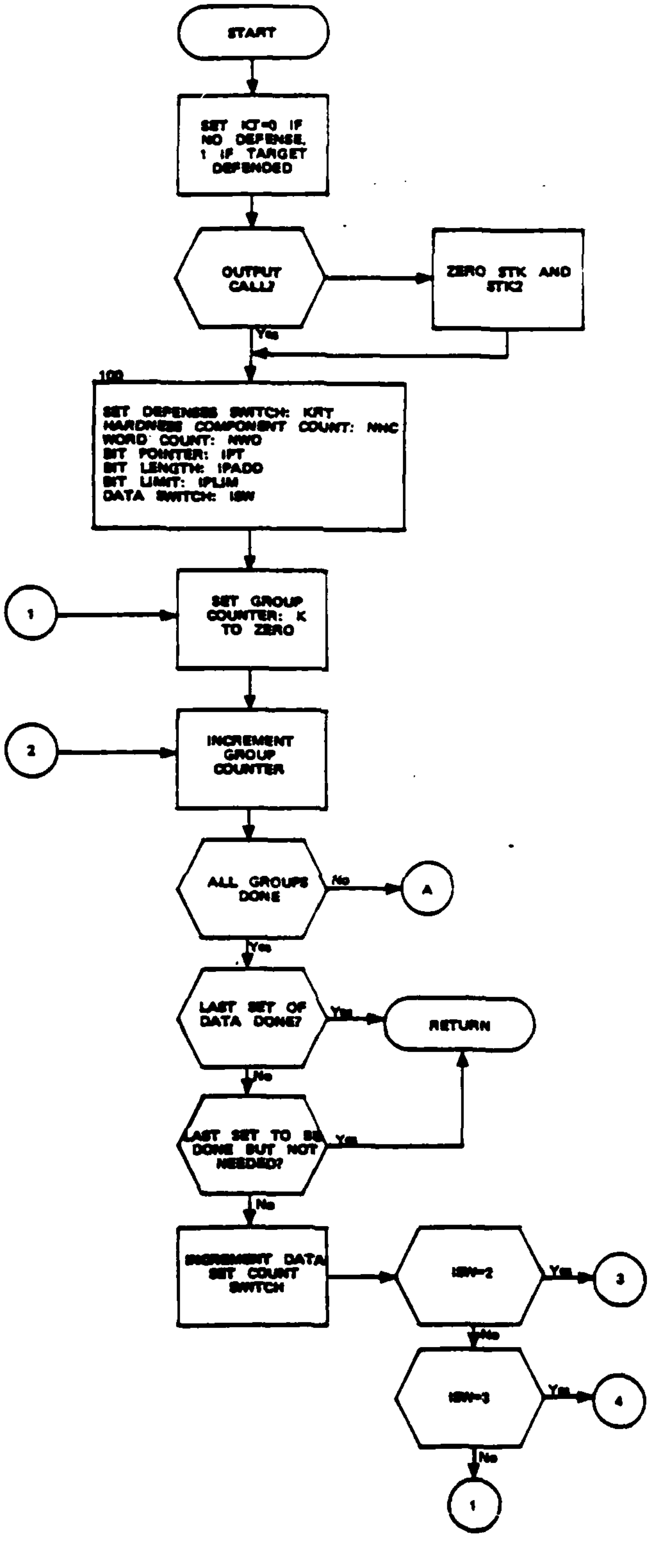


Figure 45.1. RTAPCK (Entry RTAPCK)
(Part 1 of 8)

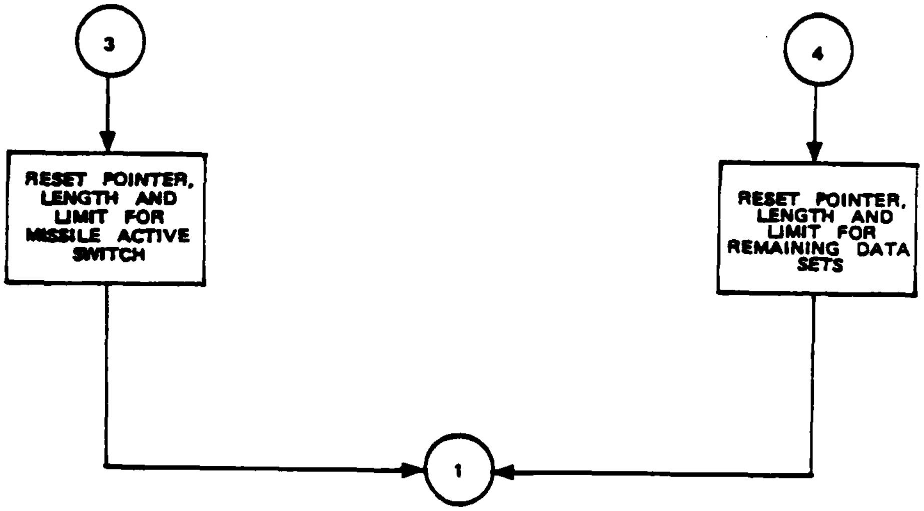

3
RESET POINTER, LENGTH AND LIMIT FOR MISSILE ACTIVE SWITCH
4
RESET POINTER, LENGTH AND LIMIT FOR REMAINING DATA SETS
1

Figure 45.1. (Part 2 of 8)

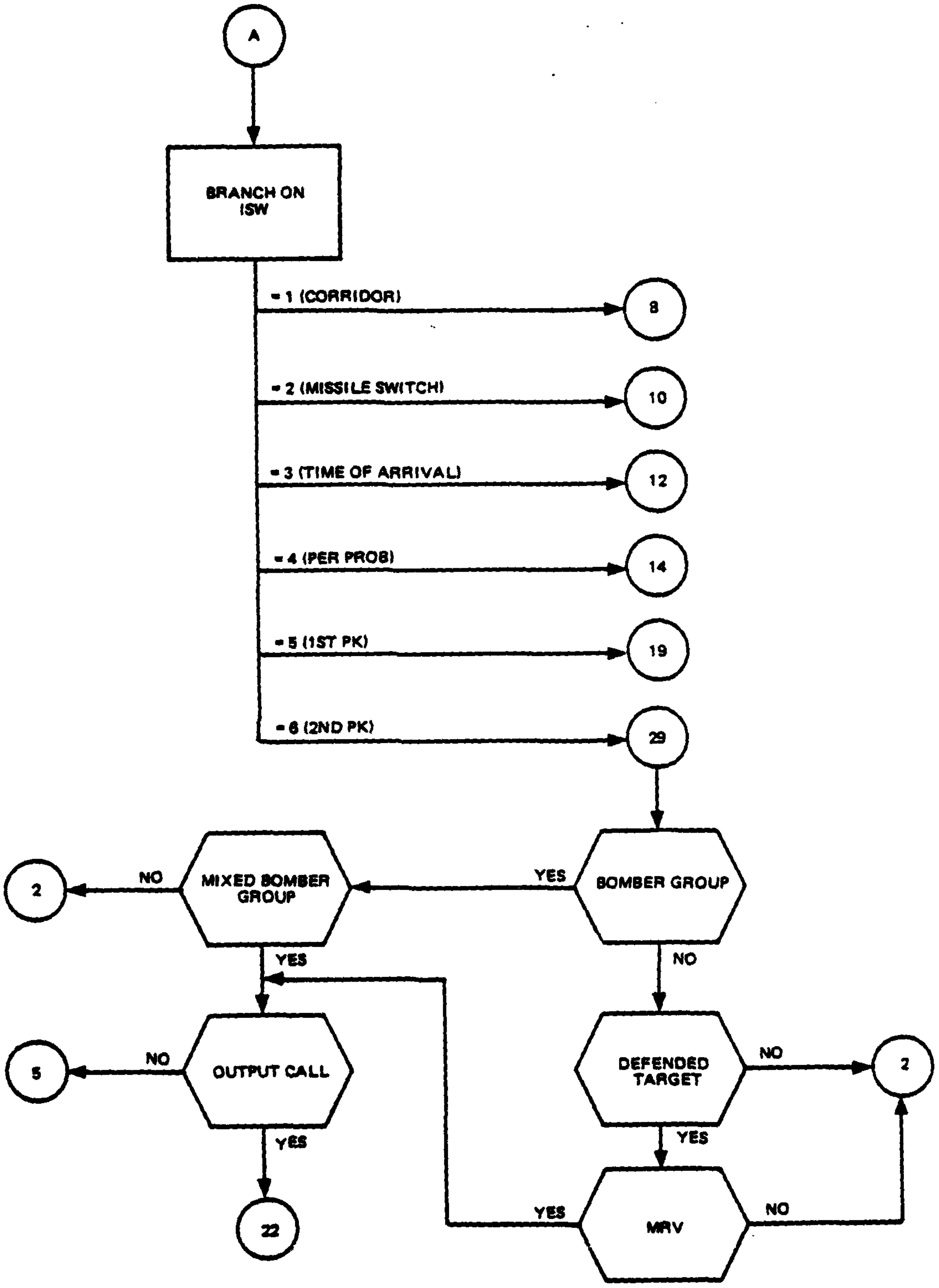


Figure 45.1. (Part 3 of 8)

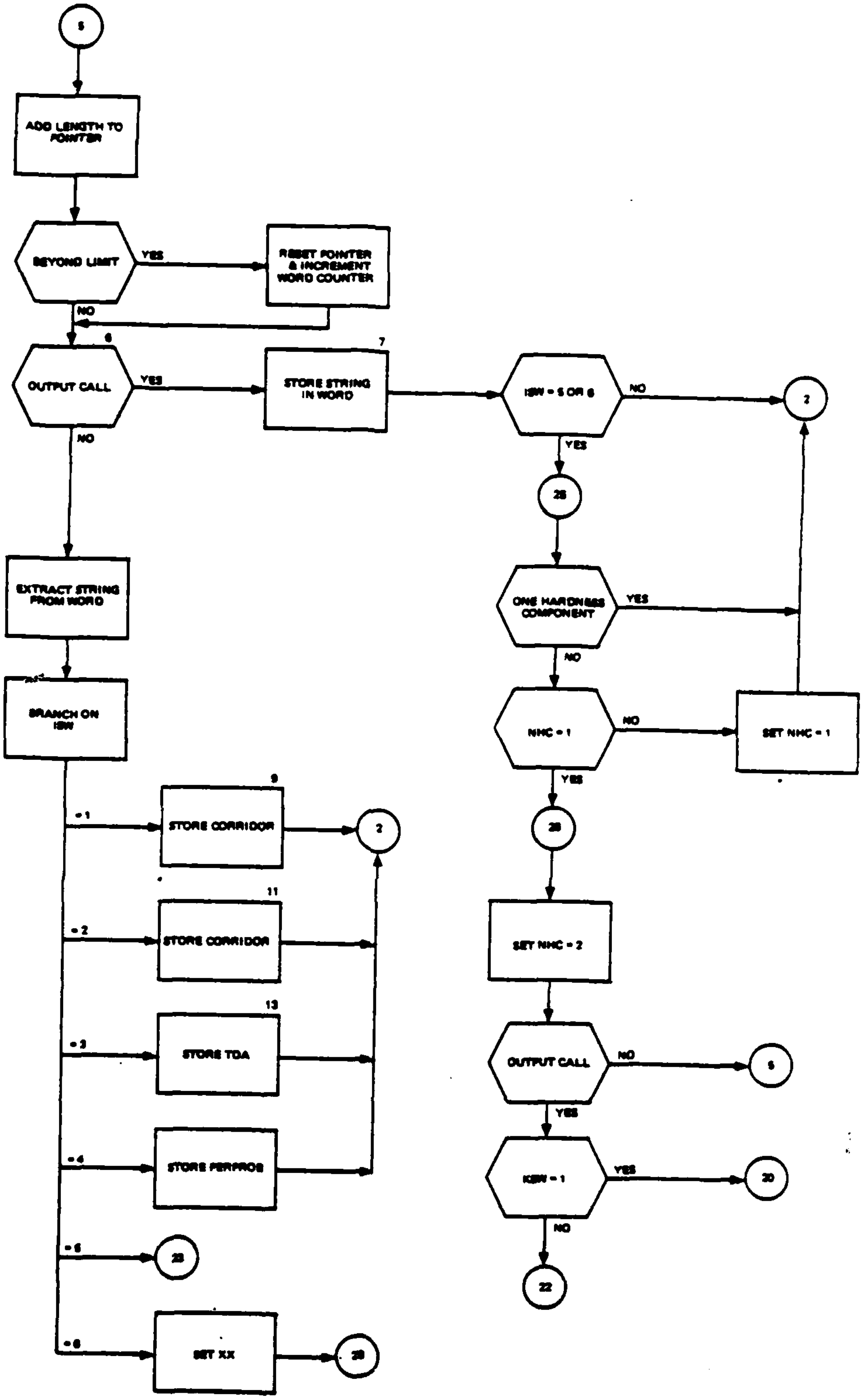


Figure 45.1. (Part 4 of 8)

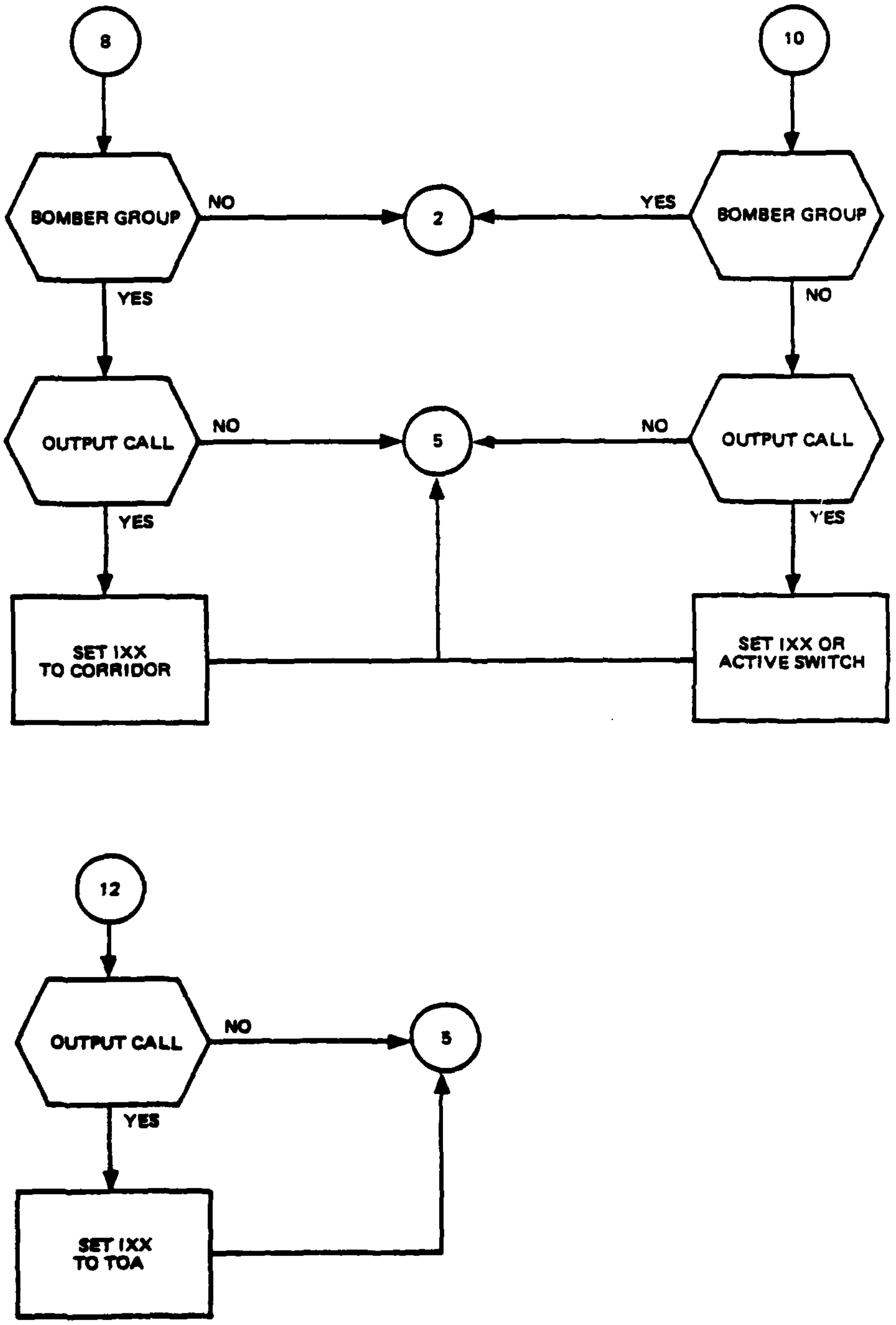


Figure 45.1. (Part 5 of 8)

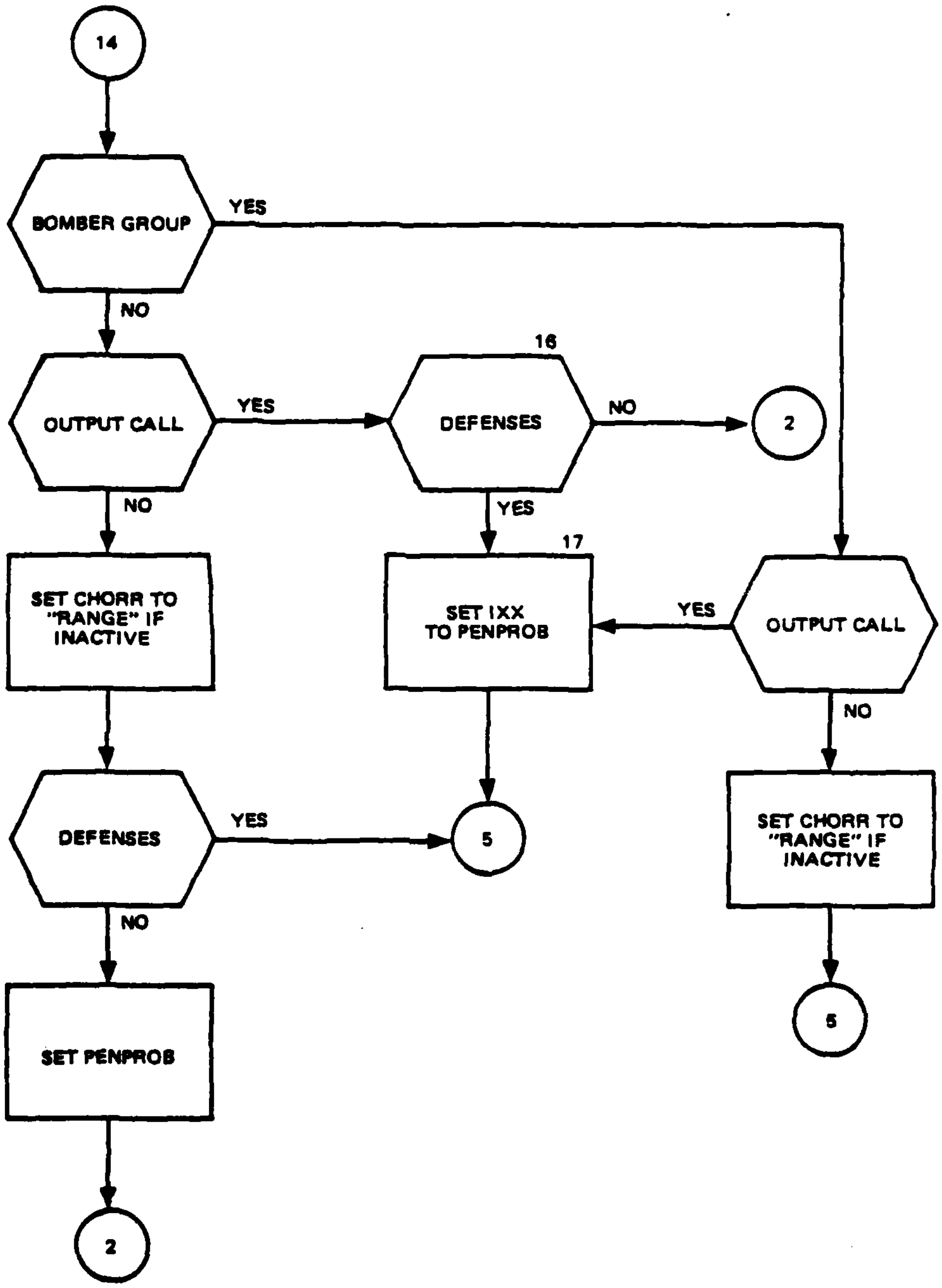


Figure 45.1. (Part 6 of 8)

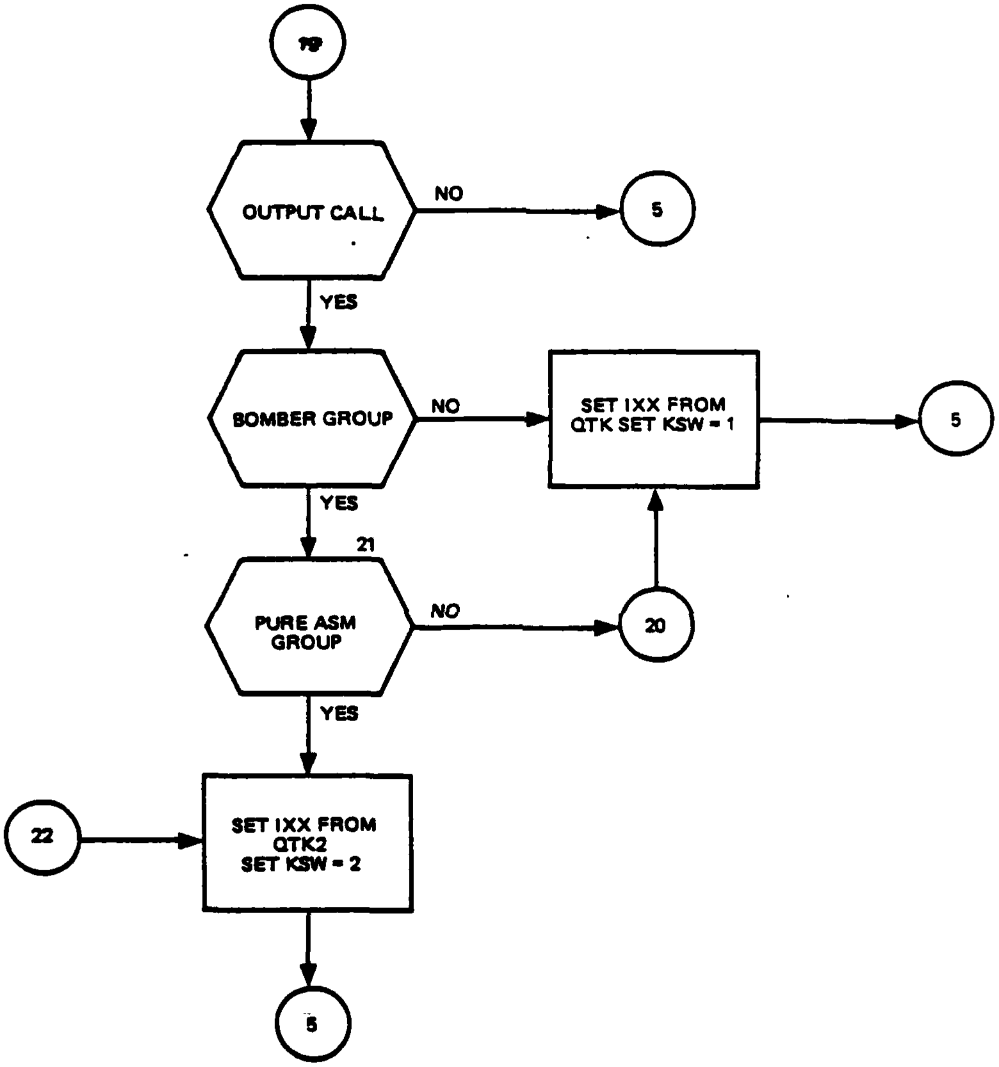


Figure 45.1. (Part 7 of 8)

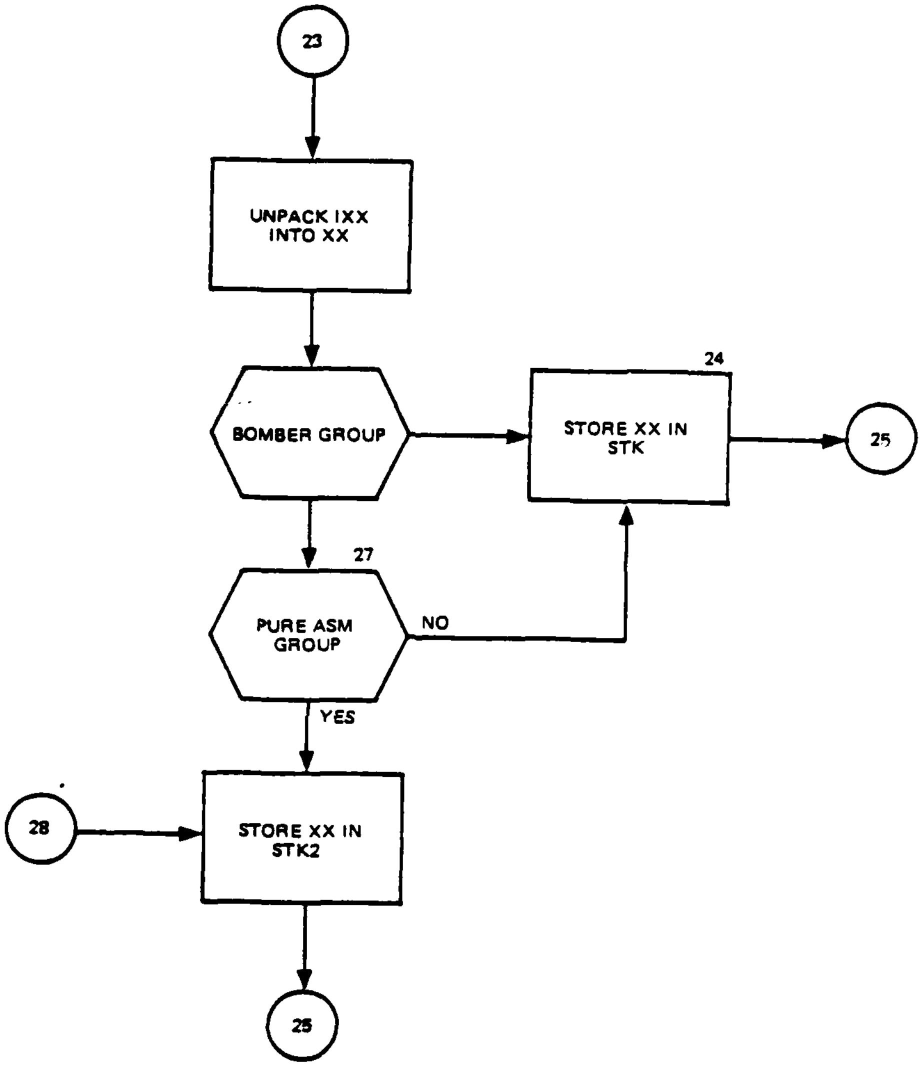


Figure 45.1. (Part 8 of 8)

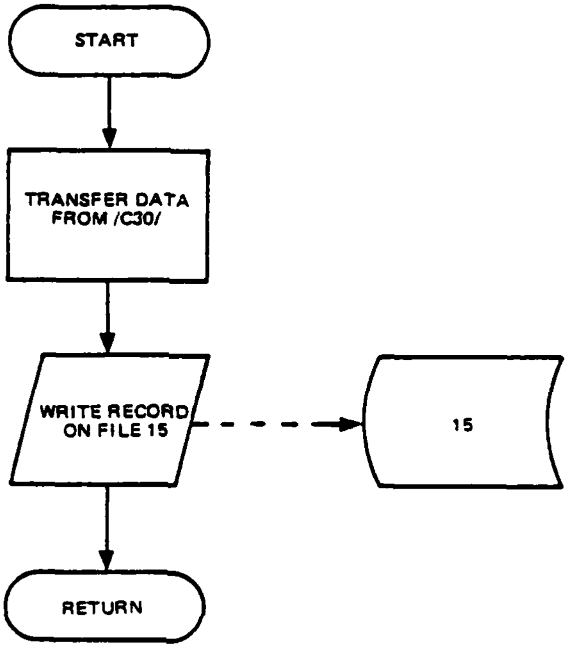


Figure 45.2. RTAPCK (Entry TGTOUT)

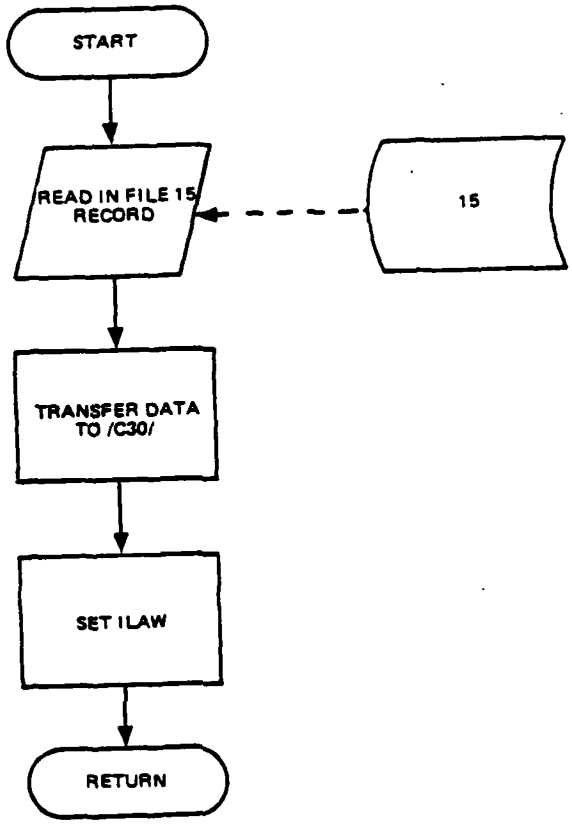


Figure 45.3. RTAPCK (Entry TGTIN)

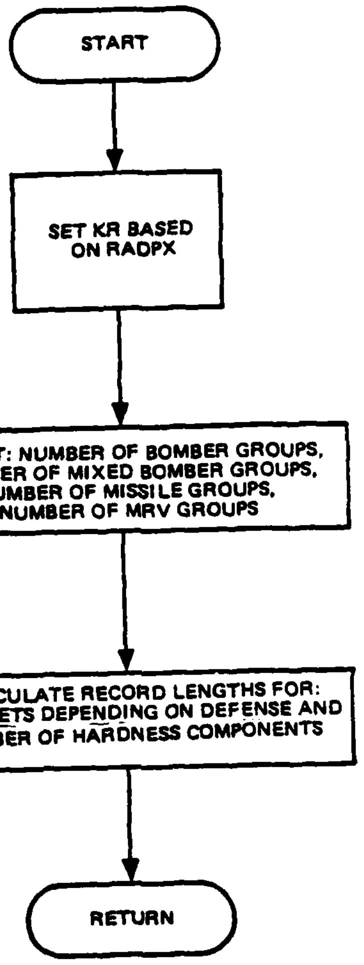

START
SET KR BASED ON RADPX
COUNT: NUMBER OF BOMBER GROUPS, NUMBER OF MIXED BOMBER GROUPS, NUMBER OF MISSILE GROUPS, NUMBER OF MRV GROUPS
CALCULATE RECORD LENGTHS FOR: TARGETS DEPENDING ON DEFENSE AND NUMBER OF HARDNESS COMPONENTS
RETURN

Figure 45.4. RTAPCK (Entry RTAPIN)

3.9.8 Subroutine SETPAY

PURPOSE: This routine sets the bomber payload indicators to specify use of gravity bombs or ASMs.

ENTRY POINT: SETPAY

FORMAL PARAMETERS: None

COMMON BLOCKS: C30, CONTRO, DYNAMI, PAYSAV, SALVO, TGTSAV, WADWPN, WEPSAV

SUBROUTINES CALLED: None

CALLED BY: RECON

Method:

This subroutine sets the bomber payload indicators (array ISETPAY in common block /SALVO/). The setting calculation considers the allocation rate of ASMs (array FASM in /SALVO/), the actual fraction of ASMs in each group array GSEASM in /PAYSAV/, the damage difference between the weapons, the average damage difference (array AVDE of /SALVO/), and the state of allocation progress (variable PROGRESS in /CONTRO/).

Five local variables are of some significance to the calculation. The variable IPREF is used for temporary storage of the payload indicators. A zero value specifies use of a bomb; a value of one specifies use of an ASM. The variable DEA contains the expected damage (ignoring the same vehicle planning factors) if a bomb were used. Local variable CONPAY is used to weigh the difference in allocation rates relative to the difference in damage expectancies. Local variable PAYSENS is used to calculate CONPAY.

The variable CONPAY is used to reflect the importance of the allocation rate difference relative to the damage difference. As CONPAY decreases, the allocation rate difference increases in importance relative to the damage difference and vice versa. CONPAY is defined as follows:

$$\text{CONPAY} = \frac{\text{PAYSENS}}{(\text{CLOSE} + 1.5) * \text{PROGRESS}}$$

Thus, PAYSENS is merely a multiplicative factor, set internally by SETPAY to a value of 0.1 which provides a base value for CONPAY. The denominator of the right-hand term of the equation represents the effects of allocation progress. As the allocation proceeds, CONPAY decreases to put more weight on the allocation rate differences. Variable CLOSE is a user input parameter which determines the size of the closing forces. When PROGRESS equals one, CLOSE increases to increase the closing forces. The table below displays the values of the denominator for different values of PROGRESS. CLOSE is set to its default value, 1.05, and CLOSER (another input parameter) is set to its default value, 4.

Table CONPAY Denominator vs. PROGRESS

Denominator	PROGRESS
1.02	.4
1.275	.5
1.913	.75
2.55	1.00 (initial)
6.55	1.00 (after one pass)
8.55	1.00 (highest value of denominator)

After PROGRESS equals one, the value of CLOSE increases by an amount CLOSER for each pass at PROGRESS equal one. The maximum number of passes with this value of PROGRESS is 1.5.

When PROGRESS is zero or in a verification pass, the weapon with the greater damage is selected for allocation. Also, the selection equations are bypassed if a group has only bombs or ASMs.

Figure 46 displays the logic or subroutine SETPAY.

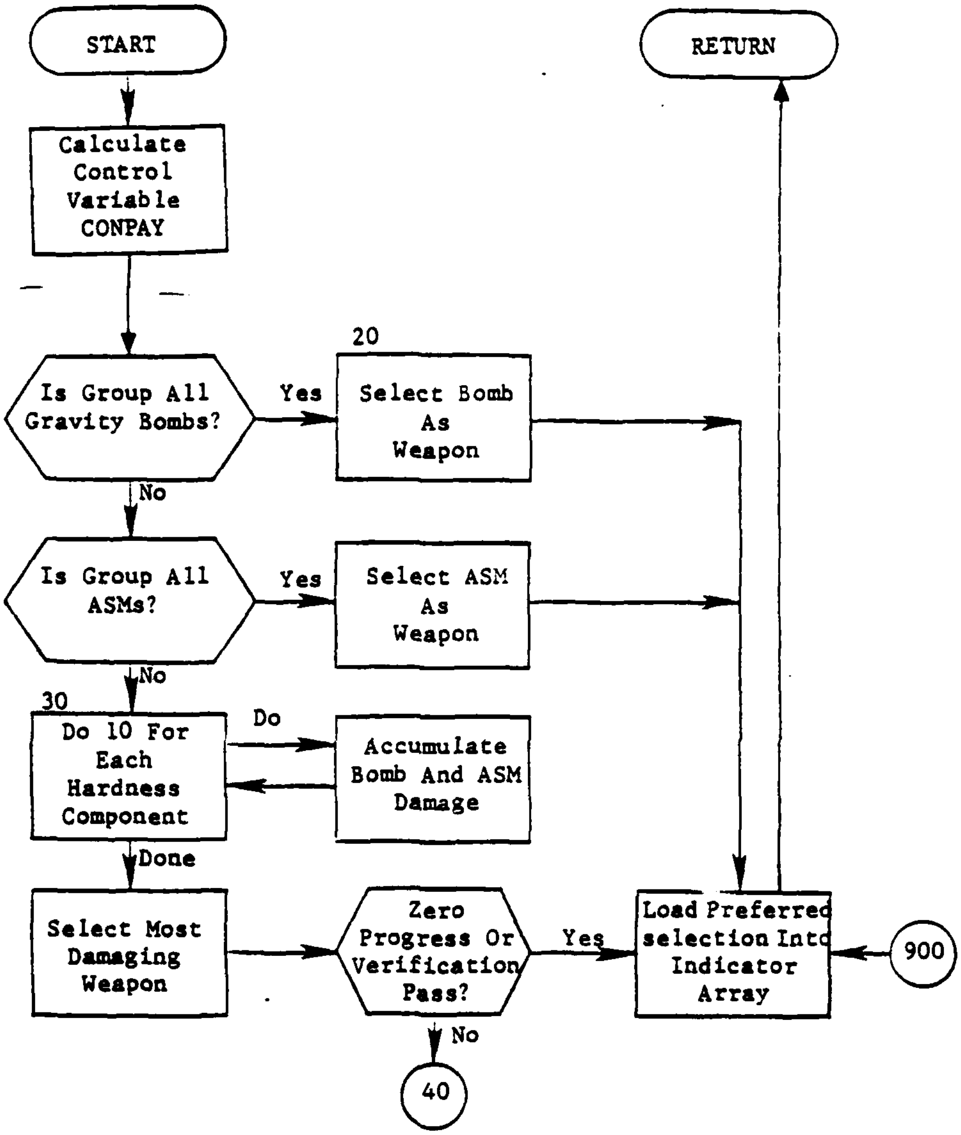

START
RETURN
Calculate Control Variable CONPAY
20
Is Group All Gravity Bombs?
Yes
Select Bomb As Weapon
No
Is Group All ASMs?
Yes
Select ASM As Weapon
No
30
Do 10 For Each Hardness Component
Do
Accumulate Bomb And ASM Damage
Done
Select Most Damaging Weapon
Zero Progress Or Verification Pass?
Yes
Load Preferred selection Into Indicator Array
900
No
40

Figure 46. Subroutine SETPAY (Part 1 of 2)

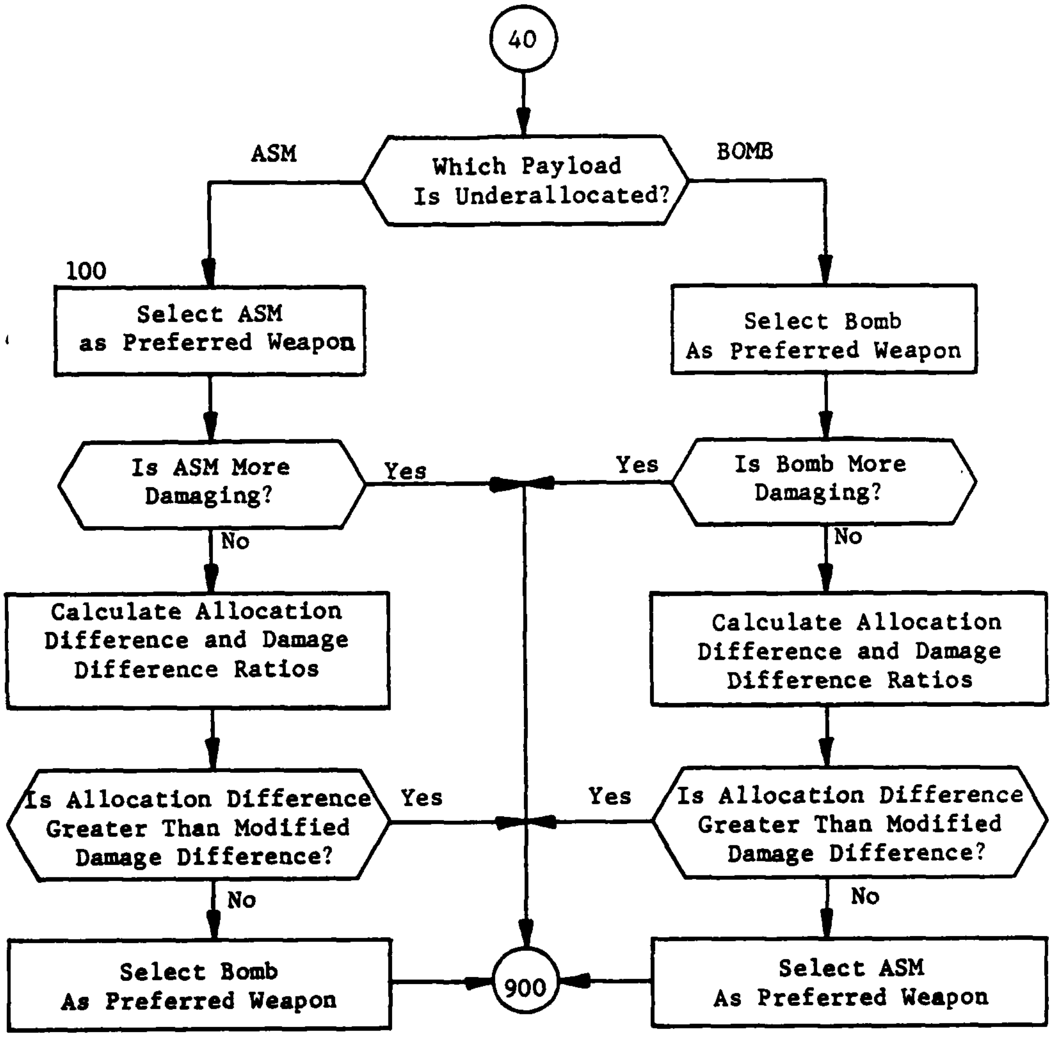

40
Which Payload Is Underallocated?
ASM
BOMB
100
Select ASM as Preferred Weapon
Select Bomb As Preferred Weapon
Is ASM More Damaging?
Yes
Yes
Is Bomb More Damaging?
No
No
Calculate Allocation Difference and Damage Difference Ratios
Calculate Allocation Difference and Damage Difference Ratios
Is Allocation Difference Greater Than Modified Damage Difference?
Yes
Yes
Is Allocation Difference Greater Than Modified Damage Difference?
No
No
Select Bomb As Preferred Weapon
900
Select ASM As Preferred Weapon

Figure 46. (Part 2 of 2)

3.10 Subroutine SCNDGD[*]

PURPOSE: To read in data on second and later ALOC passes over the target list.

ENTRY POINTS: SCNDGD

FORMAL PARAMETERS: None

COMMON BLOCKS: C10, C30, FIRST, TARREF, WADWPN, WEPSAV, WPFIX, PNAV, IRTLEN

SUBROUTINES CALLED: BOMPRM, DIRECT, FRSPLT, HEAD, NEXTTT, RECON, TGTIN, RTAPCK, ASGIN

CALLED BY: MULCON

Method:

First, if this is the first target of the pass, units 15 and 21 and perhaps 22 are rewound. Then the next target is read and all its data accessed. Next files 21 and 15 are read in. If file 22 is in use for this target it is read with its contents replacing those of file 15. The INACTIVE array is now loaded and the contents of file 15 (22) unpacked into the WADWPN block. Subroutine ASGIN is called to read the previous assignments and store them in the DYNAMI block. Finally, BOMPRM is called to remove the assignments effects from ASM totals and RECON is called for all groups.

Subroutine SCNDGD is illustrated in figure 47.

[*] First subroutine in segment SGD -- all other routines appear in section 3.9.

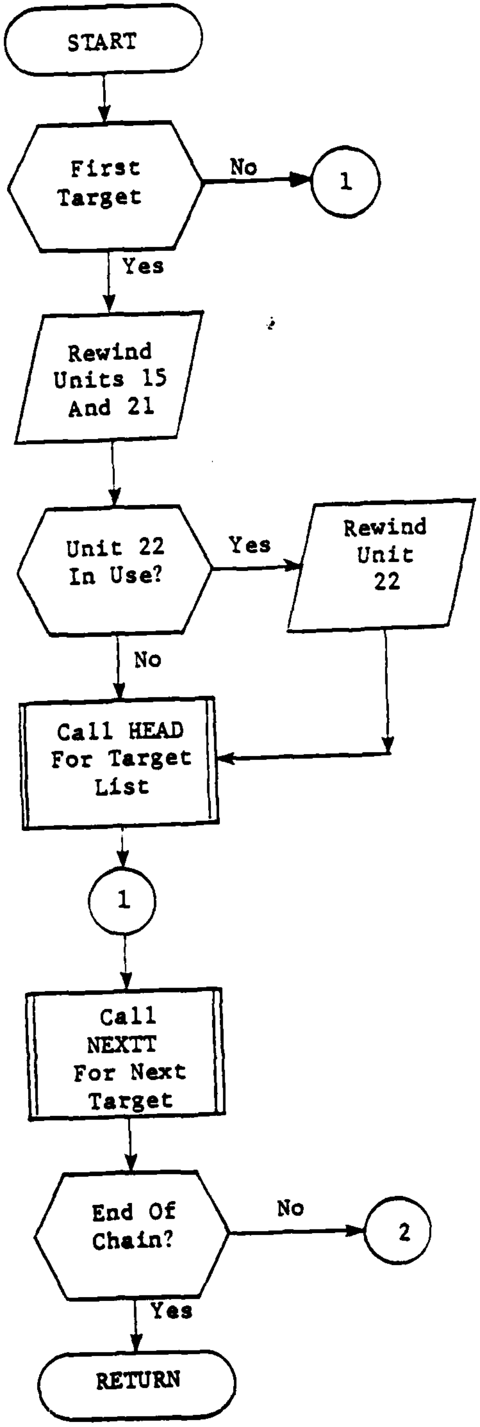


Figure 47. Subroutine SCNDGD (Part 1 of 4)

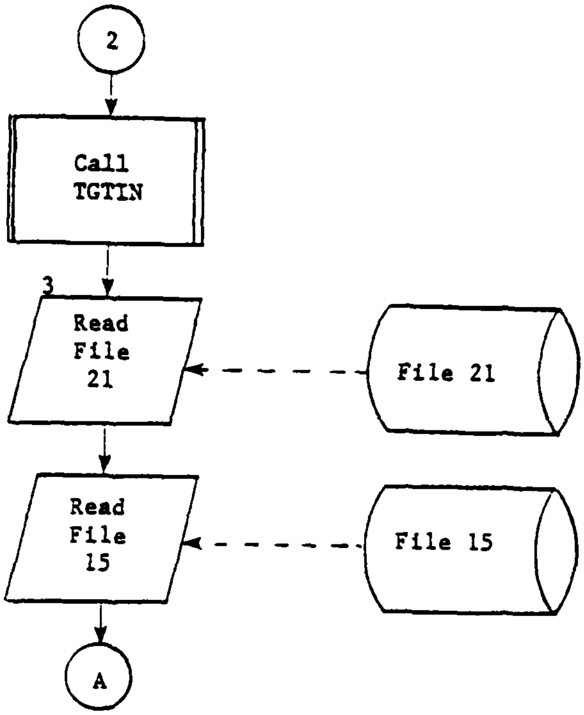


Figure 47. (Part 2 of 4)

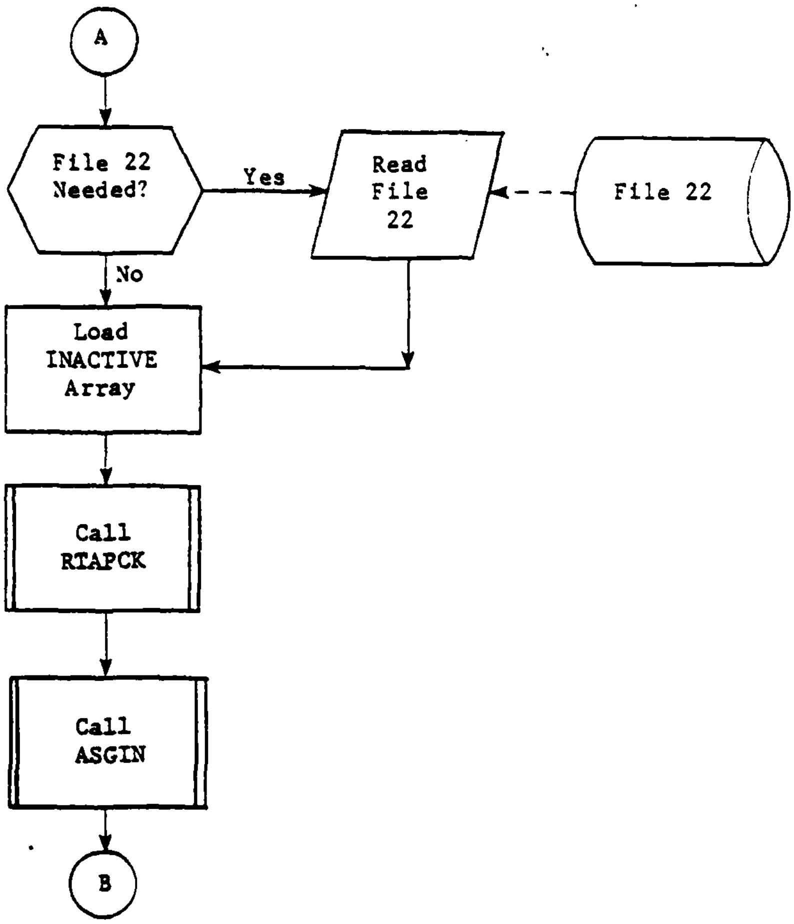


Figure 47. (Part 3 of 4)

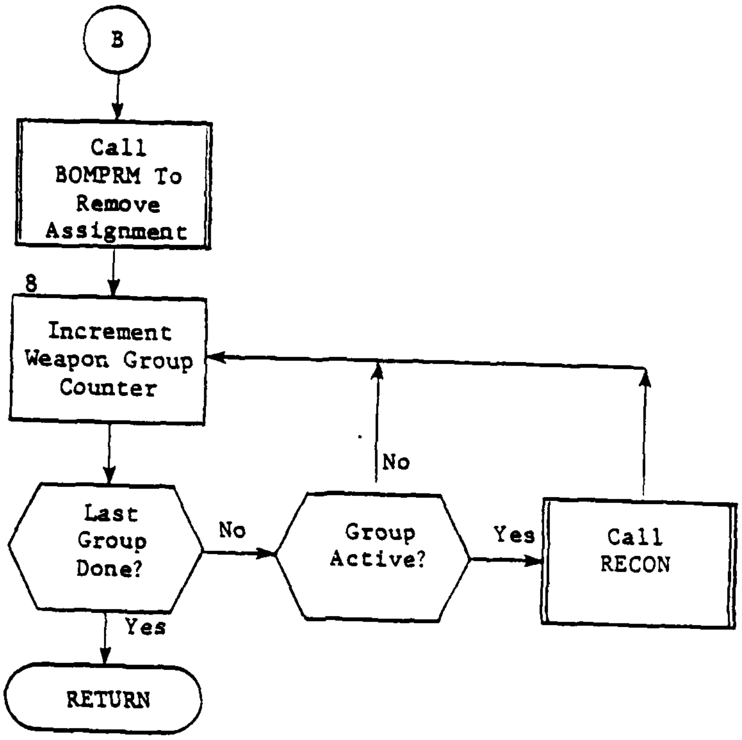

B
Call BOMPRM To Remove Assignment
8
Increment Weapon Group Counter
Last Group Done?
No
Yes
RETURN
Group Active?
No
Yes
Call RECON

Figure 47. (Part 4 of 4)

3.11 Subroutine STALL*

PURPOSE: This routine* determines the sequence of weapon additions and deletions required to achieve a near optimum allocation of weapons to a target.

ENTRY POINTS: STALL

FORMAL PARAMETERS: None

COMMON BLOCKS: C30, C33, CONTRO, DYNAMI, SALVO, SURPW, WADFIN, WADOTX, WADWPN

SUBROUTINES CALLED: ADDSAL, INITSAL, RESTORE, WAD

CALLED BY: MULCON

Method:

STALL controls WAD's addition and deletion of weapons by setting the values of the following three variables:

WADOP	in /CONTRO/
G	in /WADOTX/
NW	in /WADOTX/

WADOP has the following options:

WADOP = 1	Initialize allocation
WADOP = 2	Finalize allocation (optional print of results)
WADOP = 3	Add weapon from group (G)
WADOP = 4	Delete (NW)th weapon now on target

To facilitate monitoring the operation of STALL, the variable STALPRIN is also set to provide a unique indicator of the position in STALL where WAD is called. This variable is printed under the print option 22.

The input data for STALL consists of the following six variables supplied by WADOUT in /WADOTX/:

PPMX and IPPMX - the maximum potential increase in effective profit for any weapon, and the index G to that weapon group, respectively.

* First subroutine of segment STAL.

PVRMX and IPVRMX - the maximum effective efficiency for any weapon, and the index to that weapon group, respectively.

DPMN and IDPMN - the minimum marginal effective profit for any weapon now assigned, and the index to that weapon in the list of weapons assigned, respectively.

The flowchart for STALL is in four parts. The first part contains the setup and single weapon allocation phase. This phase provides a prompt exit from STALL if the indicated allocation consists of one weapon or less. This part also includes a dummy version of STALL which is used to reproduce the prior allocation independent of current payoff data. This option is used in place of the usual verification phase (when PROGRESS = 2) if IVERFY = 2. This mode of operation checks the effects of an alternate level of interweapon correlation, CORR2.

Before going into the normal allocation phase, the initial value of the time-of-arrival error allowance DELTVAL is saved, so that it can be restored if it is necessary to change it. This quantity determines the maximum fractional difference in target value at the time-of-arrival of weapons that are allowed to use the same time-of-arrival bin in the calculations by WAD. If the indicated allocation would result in an overflow of the available time-of-arrival bins, this quantity is increased by STALL and the allocation is reinitialized for another attempt.

Before performing any operations, STALL first calls subroutine INITSAL. This routine initializes the arrays in common block /SALVO/. At the end of the routine, subroutine RESTORE restores the multipliers for the salvoed groups.

The second part of STALL processes the fixed assignment data. It puts the weapons down on target. On the first pass, the fixed assignments were placed in array IG by FRSTGD. In later passes, they are in the IGO array. The statements after 126 determine the number of weapons the assignment represents. If DEFALOC has made an allocation on the previous pass, the number of missiles allocated from each group is shown as a negative number in the KORRX array. If the KORRX entry is positive, there is only one weapon assigned from the group. STALL then checks the INACTIVE flag to see if the weapon can reach the target. If not, an error message is printed and the assignment request is ignored. If statement 443, WAD is called to actually put the weapon on target. If the weapon is a salvoed missile, subroutine ADDSAL is called to modify the salvoed weapon stockpile.

For the fixed weapons, no change to the variable SURPWP is made as the weapon is allocated, since this variable controls the allocation of only those weapons used by the mathematical allocator.

If the fixed weapon cannot be allocated because of its active flag (INACTIVE (G)) prevents allocation, the error message notifying the user of this fact lists the reason for the inactivity. The reason is contained in array MORRX for all inactive groups.

The third part of the subroutine provides an initial laydown of weapons if multiple weapons are indicated against the target. As this laydown progresses two types of array overflows could occur. The overflow of available time-of-arrival bins has just been discussed. It results in simply restarting the allocation. The other possibility is that the total number of weapons assigned could exceed the maximum number (30) permitted. In this event, control is passed to the refinement loop just as it would be in the normal exit at the bottom of Part III.

Part IV independently checks for such a potential overflow (near statement 59) and if it is threatened, a cycle of operation is generated through the connector Ⓓ that results in removal of the least profitable weapons (statement 52) and replacement by the most profitable available weapons (statement 56). This sequence is terminated either by entering the normal refinement process when the residual target value is reduced so that no potential weapons remain profitable (statement 54), or by using exit Ⓕ after statement 59 if no combination of 30 weapons can be found which will reduce the target value sufficiently.

The operation of the normal refinement loop, which cycles through the branch at statement 71 until the tests at statement 66 are satisfied, is discussed in detail in appendix A and will not be repeated here.

Figure 48 illustrates segment STALL.

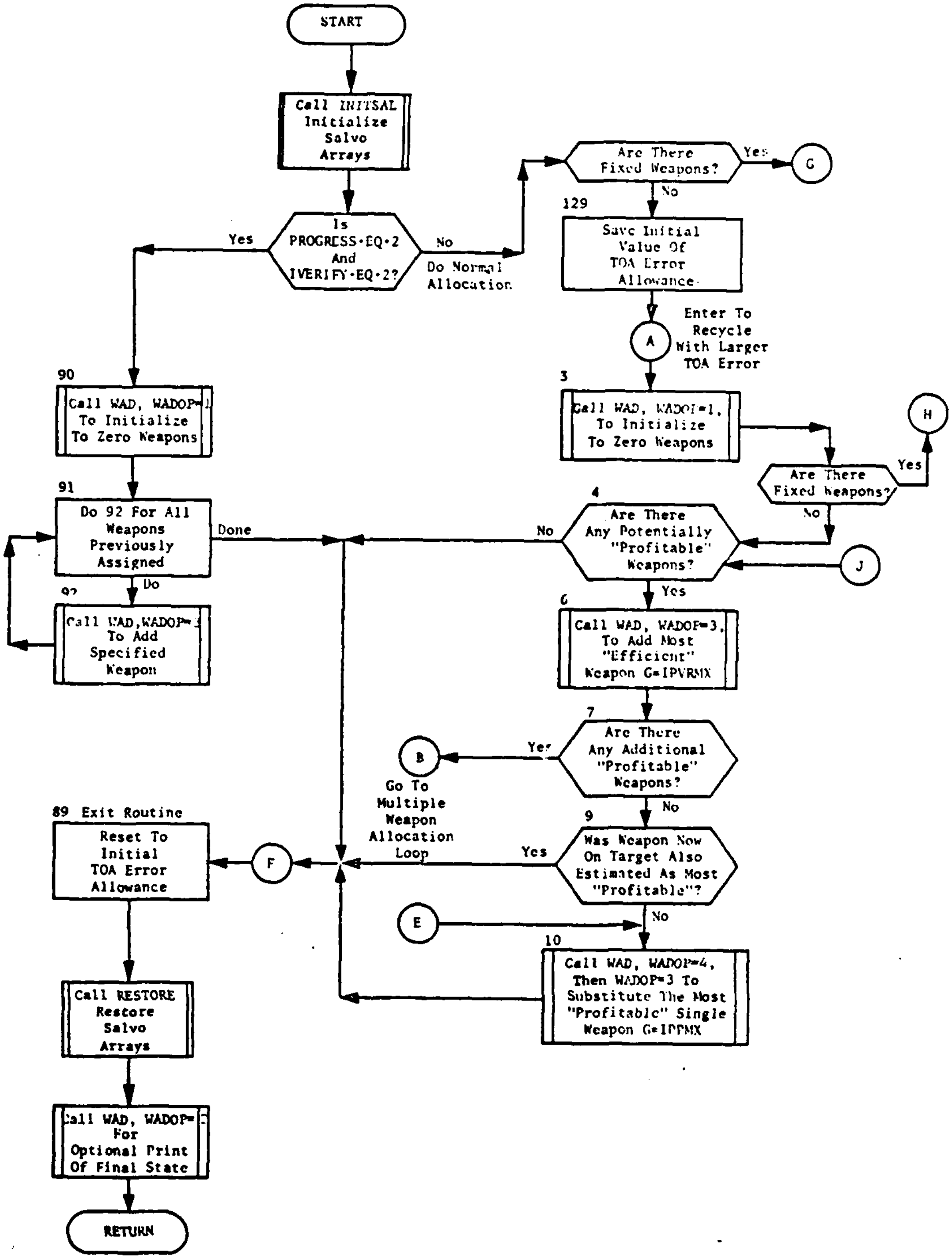

START
Call INITSAL Initialize Salvo Arrays
Is PROGRESS·EQ·2 And IVERIFY·EQ·2?
Yes
No
Do Normal Allocation
Are There Fixed Weapons?
Yes
G
No
129
Save Initial Value Of TOA Error Allowance
Enter To Recycle With Larger TOA Error
A
3
Call WAD, WADOP=1, To Initialize To Zero Weapons
H
Yes
Are There Fixed Weapons?
No
90
Call WAD, WADOP=1 To Initialize To Zero Weapons
91
Do 92 For All Weapons Previously Assigned
Done
Do
92
Call WAD,WADOP=3 To Add Specified Weapon
4
Are There Any Potentially "Profitable" Weapons?
No
J
Yes
6
Call WAD, WADOP=3, To Add Most "Efficient" Weapon G=IPVRMX
7
Are There Any Additional "Profitable" Weapons?
Yes
B
Go To Multiple Weapon Allocation Loop
No
9
Was Weapon Now On Target Also Estimated As Most "Profitable"?
Yes
89 Exit Routine
Reset To Initial TOA Error Allowance
F
E
No
10
Call WAD, WADOP=4, Then WADOP=3 To Substitute The Most "Profitable" Single Weapon G=IPPMX
Call RESTORE Restore Salvo Arrays
Call WAD, WADOP=2 For Optional Print Of Final State
RETURN

Figure 48. Segment STALL
Part I: Setup and First Weapon

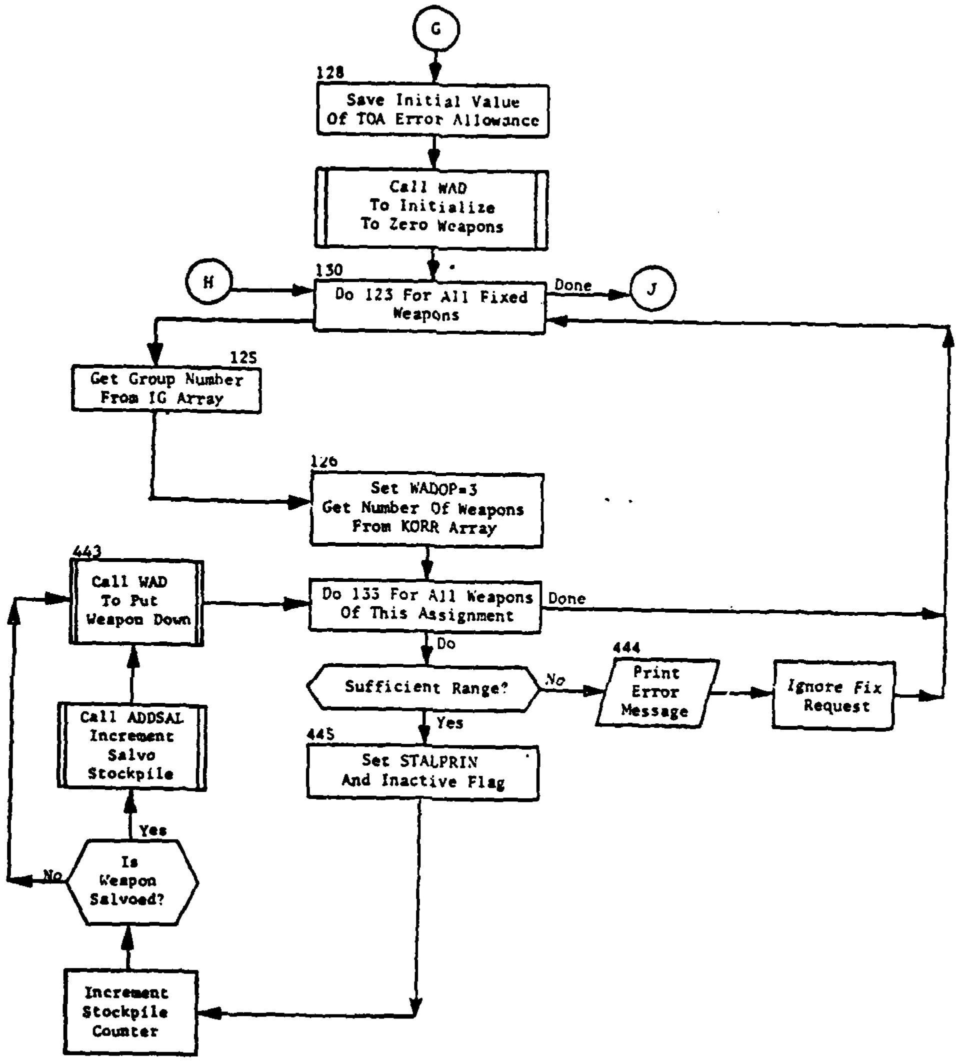

G
128
Save Initial Value Of TOA Error Allowance
Call WAD To Initialize To Zero Weapons
H
130
Do 123 For All Fixed Weapons
Done
J
125
Get Group Number From IG Array
126
Set WADOP=3 Get Number Of Weapons From KORR Array
443
Call WAD To Put Weapon Down
Do 133 For All Weapons Of This Assignment
Done
Do
Sufficient Range?
No
444
Print Error Message
Ignore Fix Request
Yes
445
Set STALPRIN And Inactive Flag
Call ADDSAL Increment Salvo Stockpile
Yes
Is Weapon Salvoed?
No
Increment Stockpile Counter

Figure 48. Part II: Fixed Weapon Assignment Processing

3.11.1 Subroutine FORMATS

PURPOSE: This subroutine determines the best 10-column BCD format for a variable.

ENTRY POINTS: FORMATS

FORMAL PARAMETERS: None

COMMON BLOCKS: FORMTT

SUBROUTINES CALLED: None

CALLED BY: PRNTOS

Method:

The variable to be formatted is INWORD, the first word in common /FORMTT/. (This word is equivalenced to WORDIN.) The best 10-column format is returned in NFORMAT, the second word of common /FORMAT/. The resulting value of NFORMAT can be used in a FORTRAN output statement such as: PRINT NFORMAT, INWORD.

The names WORDIN and INWORD are equivalenced to allow correct specification (real or integer) for any possible input. The name NFORMT is internally equivalenced to the variable name NFORMAT for convenience. Internal to FORMATS, the absolute value of the input variable is kept in INABS, equivalenced to ABSIN for type specification purposes.

To determine if a number is fixed or floating point, the first 12 bits of the absolute value are tested. If a bit is set, the number is assumed to be floating point, since all normalized floating point numbers have at least one bit set in this range. Thus, if an integer quantity greater than 16,777,216 is input, it will be treated as if it were a floating point variable. However, no variable input from PRNTNOW can have a value of that magnitude, so this restriction is never a handicap. Table 6 presents the returned formats for each range of input values.

Subroutine FORMATS is illustrated in figure 49.

Table 6. Calculated Formats for Variables

RANGE OF ABSOLUTE VALUE OF VARIABLE, X	FORMAT OF NEGATIVE	FORMAT OF POSITIVE
0 or Integer Variable	I10	I10
0 < X < 0.0001	E10.1	E10.2
0.0001 ≤ X ≤ 0.999999	F10.5	F10.5
0.999999 < X ≤ 99.9999	F10.4	F10.4
99.9999 < X ≤ 9999.99	F10.3	F10.3
9999.99 < X ≤ 999999.9	F10.2	F10.2
999999.9 < X	F10.1	F10.2

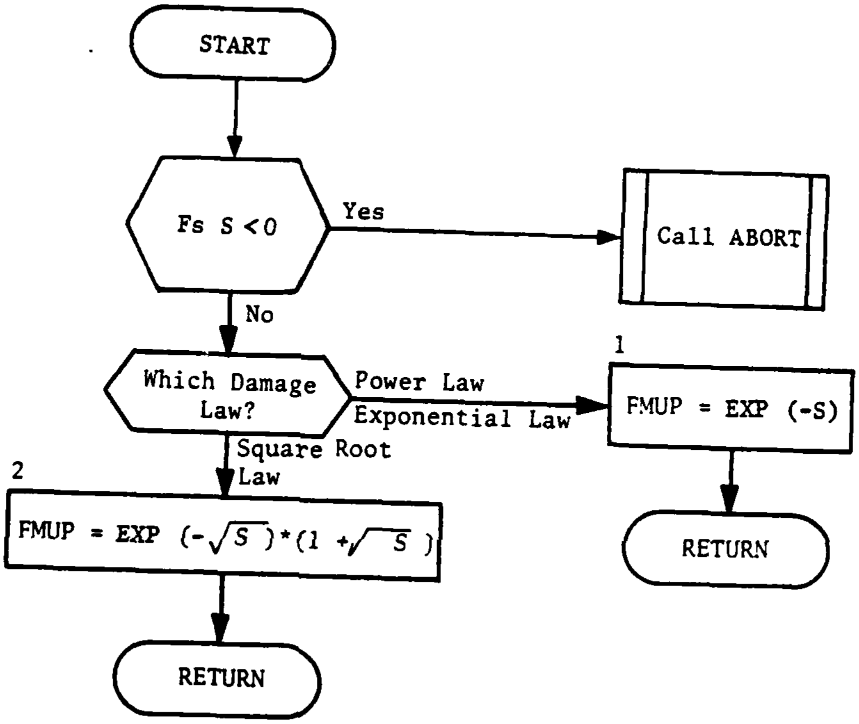


Figure 50. Function FMUP

3.11.3 Function LAMGET

PURPOSE: This real valued function calculates the Lagrange multiplier for salvoed missile groups.

ENTRY POINTS: LAMGET

FORMAT PARAMETERS:

LAM - Initial multiplier
P - Salvo balance variable
ISAL - Salvo number

COMMON BLOCKS: None

SUBROUTINES CALLED: None

CALLED BY: DEFALOC, SALVAL (entry INITSAL)

Method:

Note that this function is a real valued function. Its correct usage requires that a REAL LAMGET specification be present in the calling program.

The formal parameters specify the original or first salvo multiplier (LAM), the salvo balance variable maintained by PUPDT(P), and the salvo number (ISAL).

The returned value LAMGET is computed as follows:

LAMGET = LAM - P * (ISAL -1) * LAM

Function LAMGET is illustrated in figure 51.

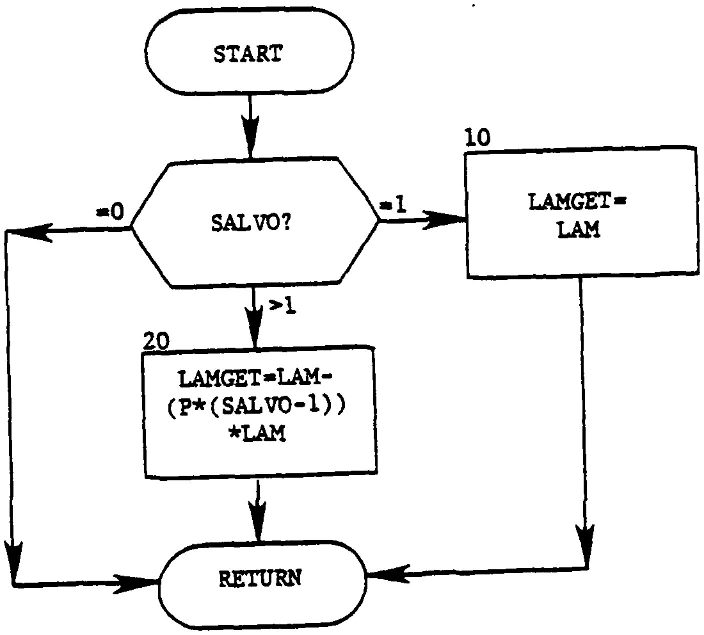


Figure 51. Function LAMGET

3.11.4 Subroutine PREMIUMS

PURPOSE: This routine calculates the premium used by WADOUT in evaluating the benefit of using or not using weapons from specific groups.

ENTRY POINTS: PREMIUMS

FORMAL PARAMETERS: G - An integer group number

COMMON BLOCKS: C30, CONTRO, PREMS, SURPW, WPFIX

SUBROUTINES CALLED: None

CALLED BY: WAD, DEFALOC, MULCON

Method:

The formal parameter G specifies for which weapon group the premium is to be recomputed. Three modes are provided for the computation of the premiums, depending on the value of PROGRESS.

PROGRESS < 1.0 A normal linear premium is computed which keeps the allocator from producing allocations with unnecessarily large deviations from the desired allocation rate (statement 12).

PROGRESS = 1.0 The normal premium is augmented by a step function which strongly motivates the allocator to exactly match the stockpile statement 1).

PROGRESS > 1.0 The subroutine exists with a zero premium for use in verification allocations (statement 10).

Since the calculation of the step function premium requires the quantity SMALLAM = .5 * LAMEF(G) for the lowest value of LAMEF, this quantity is evaluated only on the first call of PREMIUMS after PROGRESS is 1.0 (statement 18). Thereafter (since the values of LAMEF are frozen while PROGRESS = 1.0) there is no need to recompute this quantity.

PREMIUM is subtracted from the cost of adding a weapon. DPREMIUM is subtracted from the cost of deleting a weapon.

Where a weapon surplus exists (i.e., SURPWP(G) > 0), PREMIUM(G) > 0.0 and DPREMIUM(G) ≤ 0.0. Where a weapon deficiency exists (i.e., SURPWP(G) < 0), the reverse is true.

Figure 52 illustrates subroutine PREMIUMS.

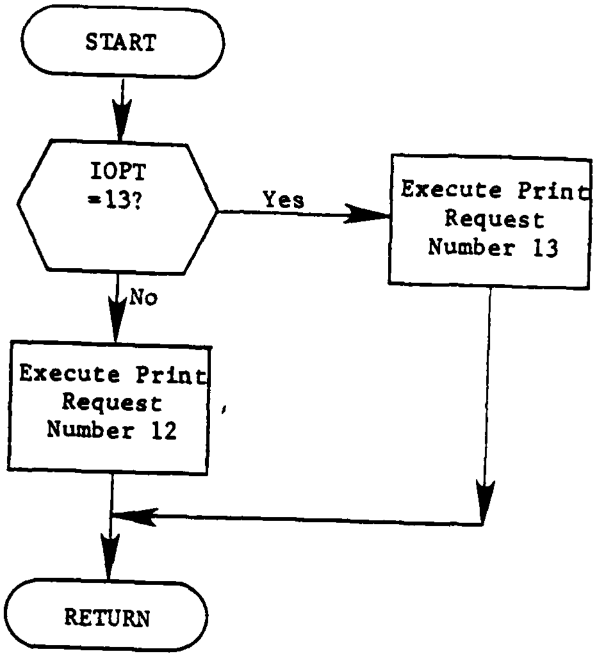

START
IOPT =13?
Yes
No
Execute Print Request Number 13
Execute Print Request Number 12
RETURN

Figure 53. Subroutine PRNTOS

3.11.6 Subroutine SALVAL

PURPOSE:	This routine selects the preferred salvo for each salvoed missile group, saves, and restores the appropriate Lagrange multipliers.
ENTRY POINTS:	SALVAL, INITSAL, NEWSAL, RESTORE
FORMAL PARAMETERS:	None
COMMON BLOCKS:	C30, CONTRO, PAYSAV, SALVO, TGTSAV, WADOTX, WADWPN, WEPSAV, WPFIX
SUBROUTINES CALLED:	GLOG, LAMGET, VALTAR
CALLED BY:	STALL, DEFALOC, WAD

Method:

There are three entry points, INITSAL, RESTORE, and NEWSAL. Entry INITSAL is called at the beginning of automatic weapon allocation in STALL or DEFALOC. First, it saves the multipliers LAM for each salvoed group in array SAVLAM. Then, for each group and each salvo, INITSAL determines the salvo number and cost of the best salvo. The profit of each salvo is defined as

$$PR_i = VTAR_i - TLAM_i$$

where $VTAR_i$ is the value of the target at arrival time of salvo i and $TLAM_i$ is the value of the multiplier for salvo i as determined by function LAMGET. The salvo with the highest PR_i is selected as the best salvo. The salvo number is entered in array MYSAL in common block /SALVO/ and the cost L_i replaces the multiplier for the group LAM. Entry RESTORE merely restores the original values of the multipliers (SAVLAM) back to the multiplier array LAM. Entry NEWSAL checks the current allocation and running sum for the weapon just added or deleted. If the salvo has been completely allocated, NEWSAL flags the salvo as unavailable. Entry SALVAL is never used.

Entry INITSAL

This entry is the most complex of SALVAL. The local variable IAM is set to "INITSL" to flag the exit points after statement 420 and at statement 500. The first processing (DO loop to statement 10) saves the Lagrange multipliers (LAM) in the SAVLAM array in /SALVO/. The major processing in INITSAL occurs in the DO loop to statement 100 over all the salvoed groups. Within this loop, the DO loop to statement 300 investigates each salvo to determine its worth. Array IHAVE in /SALVO/ is used to exclude salvos from consideration. If IHAVE (I, J) is false, then salvo I in group J does not have any available weapons and is ignored. If weapons were available, a jump is made to statement 420 in entry

NEWSAL. That section of code, described later, determines if the salvo to be considered is overallocated beyond its limit. If so, the salvo is not further considered. If the salvo is available, function VALTAR is used to obtain VTAR the target value for the salvo. Function LAMGET is used to calculate TLAM, the multiplier for the salvo. The best salvo is the one with the highest positive difference between VTAR and TLAM. When this maximum is selected, MYSAL in /SALVO/ is set to the best salvo number. LAM in /WPFIX/ is set to the appropriate multiplier. The arrays TOA, TVALTOA, and VTOA (in /WADWPN/) are set to correspond to the arrival time of the best salvo.

Entry INITSAL is illustrated in part 1 of figure 54.

Entry RESTORE

This simple entry point uses a DO loop to statement 1000 to restore the original values of the Lagrange multipliers (SAVLAM) to the multiplier array (LAM).

This entry is illustrated in part 3 of figure 54.

Entry NEWSAL

This entry is used after each allocation of salvoed weapons to determine if the salvo is still available. Each salvo has a maximum limit of overallocation. Before PROGRESS equals one, this limit is +225 which is the largest number which can be stored in the NSALAL array. (A description of the structure of the NSALAL array is contained in the Method section of subroutine ADDSAL, in this chapter.) When PROGRESS equals one, a more severe limit is imposed in order to accelerate closure to the stockpile. In this case the limit is zero. That is, a salvo is available only if it is underallocated.

To exit from this entry, local variable IAM is checked. If it is "INITSL," then the original call was to INITSAL and control passes to statement 220 in entry INITSAL. If it is "NEWSAL" the routine exits.

Note that if a salvo is unavailable because of limit on a call to NEWSAL, the routine attempts to find the closest lower available salvo. If none can be found, then MYSAL is set negative to flag that no salvo is available for this group.

Entry NEWSAL is illustrated in part 4 of figure 54.

* In line function IHAVE extracts information from logical array LXIHAVE.

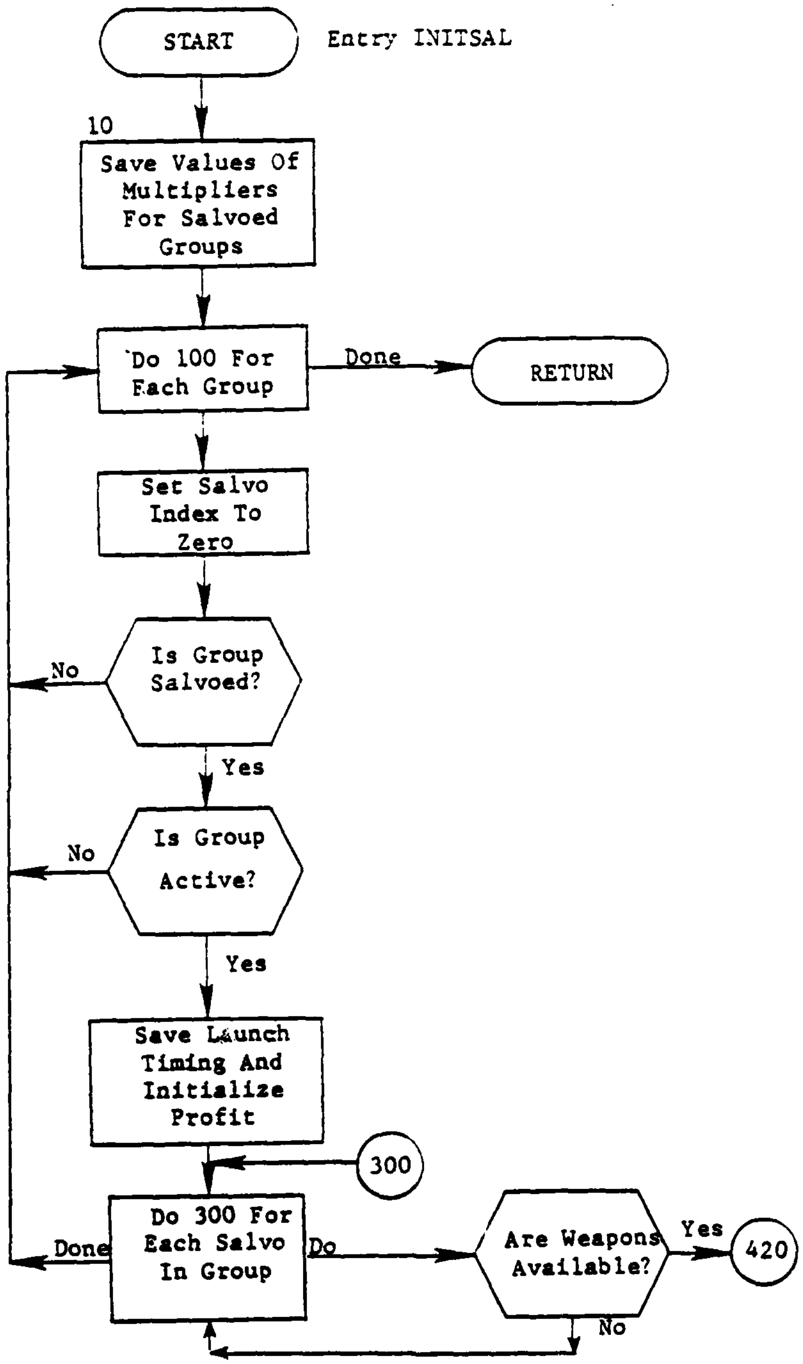


Figure 54. Subroutine SALVAL
(Part 1 of 5: Entry INITSAL)

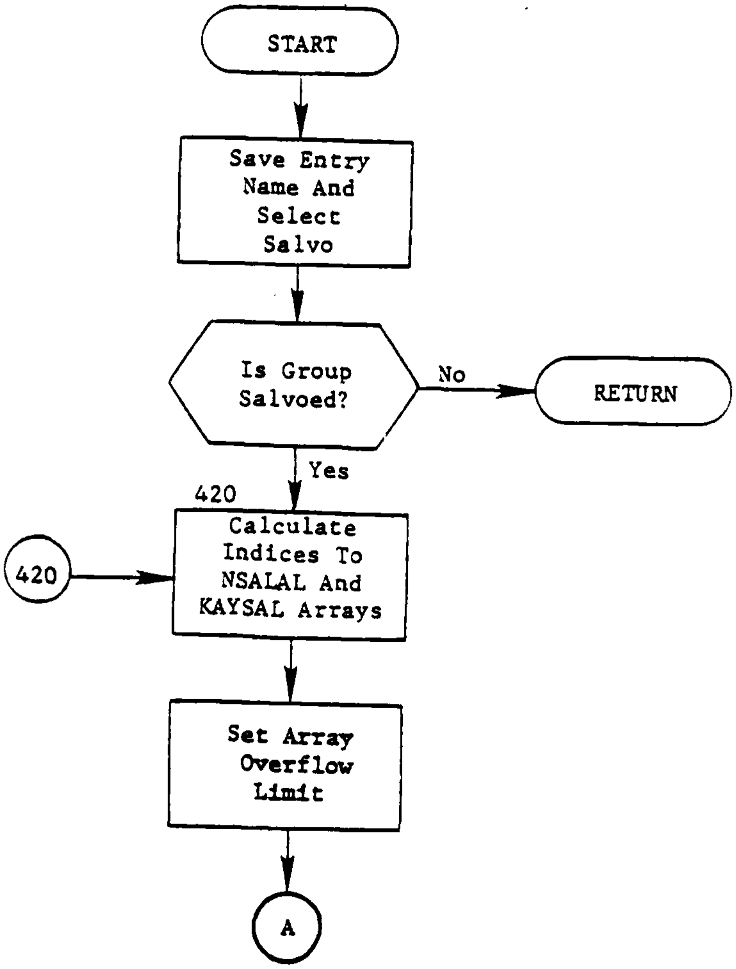

START
Save Entry Name And Select Salvo
Is Group Salvoed?
No
RETURN
Yes
420
Calculate Indices To NSALAL And KAYSAL Arrays
420
Set Array Overflow Limit
A

Figure 54. (Part 4 of 5: Entry NEWSAL)

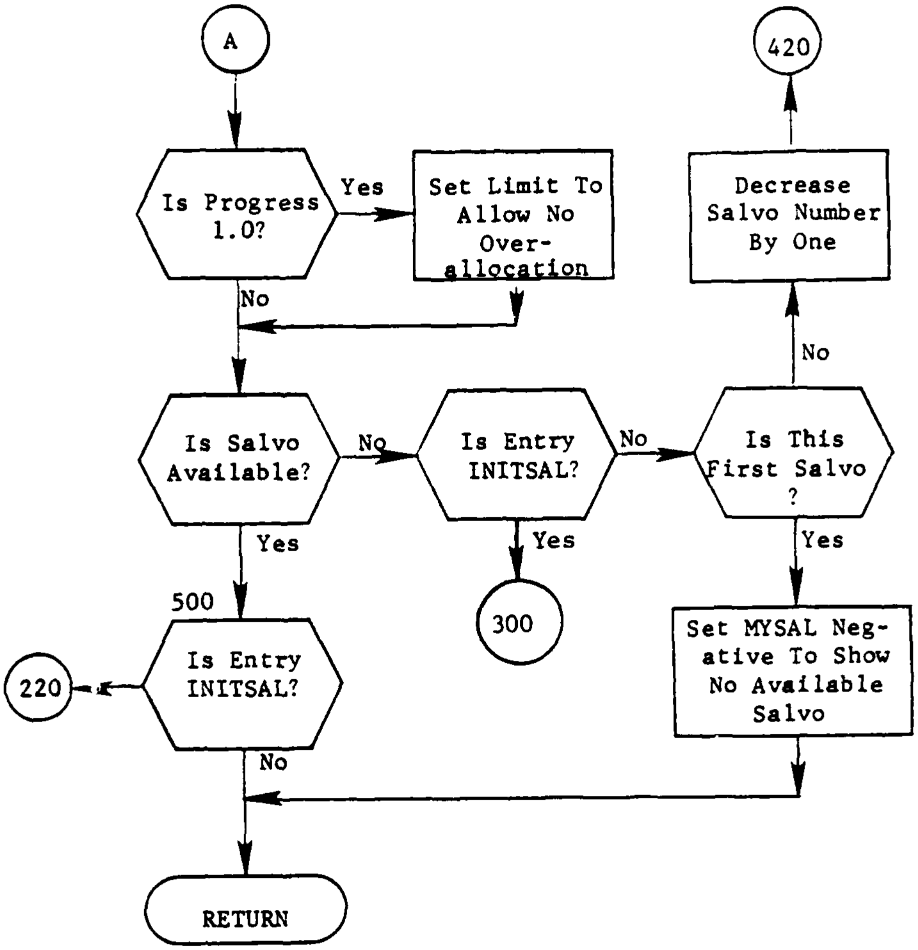


Figure 54. (Part 5 of 5)

CONTENTS OF PAGES 275 THROUGH 277 INTENTIONALLY DELETED

3.11.8 Subroutine WAD

PURPOSE: This routine carries out the addition and deletion of weapons as specified by STALL. After each change in the allocation to a target WAD computes the surviving target value VT; for each potential weapon G, the potential surviving value VTP(G) if a weapon for that group were added; and for each weapon currently on the target, the potential surviving value VTD(NW) if a weapon from that group were deleted.

ENTRY POINTS: WAD

FORMAL PARAMETERS: None

COMMON BLOCKS: C30, C33, CONTRO, DYNAMI, PREMS, SALVO, SURPW, TGTSAV, WADFIN, WADLOC, WADOTX, WADWPN, WEPSAV, WPFIX

SUBROUTINES CALLED: ADDSAL, FMUP, NEWSAL, PREMIUMS, PRNTALL, WADOUT

CALLED BY: STALL

Method:

The surviving target value VT is given by:

$$VT = \sum_{J=1}^{J=M} \sum_{N=0}^{N=NN} \left[V(N,J) - V(N+1,J)\right] * S(N,J)$$

where

$$S(N,J) = FMUP\left[(MU(N,J) ** 2)/(MU(N,J) + SIG(N,J))\right]$$

The function FMUP(X) is defined as follows:

Exponential damage law:

$$S(N,J) = FMUP(X) = \exp(-X).$$

Square root damage law:

$$S(N,J) = FMUP(X) = (1 + \sqrt{X})(\exp(-\sqrt{X})).$$

The index J is over the hardness components, and the index N is over the time of arrival bins. V(N,J) is the unattrited value of the Jth hardness component at the time corresponding to the Nth time-of-arrival bin.

However, the value of VS would be changed since the value of V(N + 1,J) should now reflect the target value for the new column 3 and would be higher than the 32.0 now shown for column 3,JH = 1. The value of MU and SIG in the new column 3 would be the same as that in column 2 except that MU would be augmented by MUP(G,J) for the new weapon and SIG would be augmented by the cross term between the new weapon and all other weapons on the target at that time. The same rule of course applies for all succeeding time-of-arrival bins. In each following column (including old column 3, now 4) the previous value of MU is increased by MUP for the weapon added, and the value of SIG is increased by the cross terms between all weapons previously on the target and the weapon added.

Of course, the new weapon might fit into one of the existing time-of-arrival bins. In this case it would be unnecessary to make a new column, and it would be unnecessary to recompute VS for the previous column. The value of MU and SIG would simply be augmented in the corresponding column and in all succeeding columns as before. Naturally, after the values of MU and SIG are revised, the value of S must be recomputed and the VS and VSN must be revised in the columns affected.

We now recall that WAD is required to provide potential weapon added target value VTP(G) for every weapon group each time a weapon is added or deleted. Obviously the calculation every time of all the above cross terms could be very time consuming. Moreover, precalculation of all individual cross terms for 250 weapon groups would be equally impractical. The technique adopted, therefore, was to calculate cross terms for each weapon group, but only with the weapons already on the target. Since the process of augmenting the values of SIG must be carried out individually for each time-of-arrival bin, the resulting data are stored by time-of-arrival bins. That is, for each weapon group G and each time-of-arrival column, N, data are stored which indicates the amount by which SIG would be increased in that column if the weapon G were added. These data are stored in the array SIGP(G,J,N). Using these data, the effective augmentation of SIG to calculate VTP(G) for each weapon group G can be accomplished simply by adding SIGP for the appropriate column, and the augmentation of MU is accomplished simply be adding MUP for each weapon G.

Table 8-A may help to visualize how these data are used. Each potential group G is tagged with the index ITOA(G) of the time of arrival column it would occupy if it were added. In addition, it is also noted whether the weapon would generate a new column (IADDTOA=1) or share the column with the weapons already there (IADDTOA=0). The situation illustrated here in table 8 corresponds to the same one illustrated in table 7. Notice that for each weapon group G the array SIGP(G,J,NI) contains a significant (usually nonzero) data for NI>ITOA(G) -IADDTOA(G). The extra term in the column NI = ITOA(G) -1 for rows 1 and 5, where the weapon group would require a new column (IADDTOA=1), is to provide a term required for the new columns if this weapon were added. Since addition of this weapon would move all columns (including its own) one position to the right, the resulting term would be in the proper position after moving, even though it is in an incorrect column at present.

Table 8. Illustrating Quantities Calculated for Potential Weapon Added and Deleted Payoffs

A. Data on All Potential Weapons

G	VTP (G)	ITOA (G)	IADDTOA (G)	J	DSIG,G,3	NI=1	NI=2	NI=3	NI=4
1		2	0	1	.021	0.0	.042	.063	
				2	.103	0.0	.206	.659	
2		3	0	1	.052	0.0	0.0	.45	
				2	.660	0.0	0.0	4.23	
3		2	0	1	.274	0.0	.548	1.653	
				2	.780	0.0	1.560	1.880	
4		3	0	1	.005	0.0	0.0	.010	
				2	.020	0.0	0.0	.040	
5		4	1	1	.003	0.0	0.0	.007	
				2	.009	0.0	0.0	.018	

B. Data on Weapons Now on Target
(As Candidates for Possible Deletion)

				SIGD(NW,J,NI)			
NW	IG(NW)	VTD(NW)	J	NI=1	NI=2	NI=3	NI=4
1	2		1	0.0	0.0	- .105	
			2	0.0	0.0	-1.32	
2	3		1	0.0	- .274	- .052	
			2	0.0	-1.560	- .660	
3	3		1	0.0	- .274	- .052	
			2	0.0	-1.560	- .660	

controlled by one of the control programs depending on the WADOP option chosen (initialize, add, or delete). The following three parts each illustrate the operation of one of these control programs. Each control routine utilizes a number of other local subroutines, as well as external routines (see figure 56).

The Add Weapon Control routine (Part III) will be used as a vehicle to illustrate the operation of the program. Once the operations of this routine are understood, the corresponding operations in the other routines should be obvious.

The routine first checks to be sure that the number of weapon additions and weapon deletion operations, IOP, on this target does not exceed 100. If it does, it is assumed that STALL and WAD are caught in an endless loop probably repeatedly adding and deleting the same weapon, so the processing of the target is terminated. In principle, such looping should not occur. However, it has been found that errors in reading file data for one target, or a random machine malfunction, or inconsistencies in the data supplied to the program can sometimes result in such a situation. This makes it possible for the program to proceed to the next target rather than aborting the entire run. However, when this happens, the LOOP flag is set nonzero. This causes the print in statement 41 to appear during the initialization of every succeeding target, so that the user is sure to notice that the difficulty occurred.

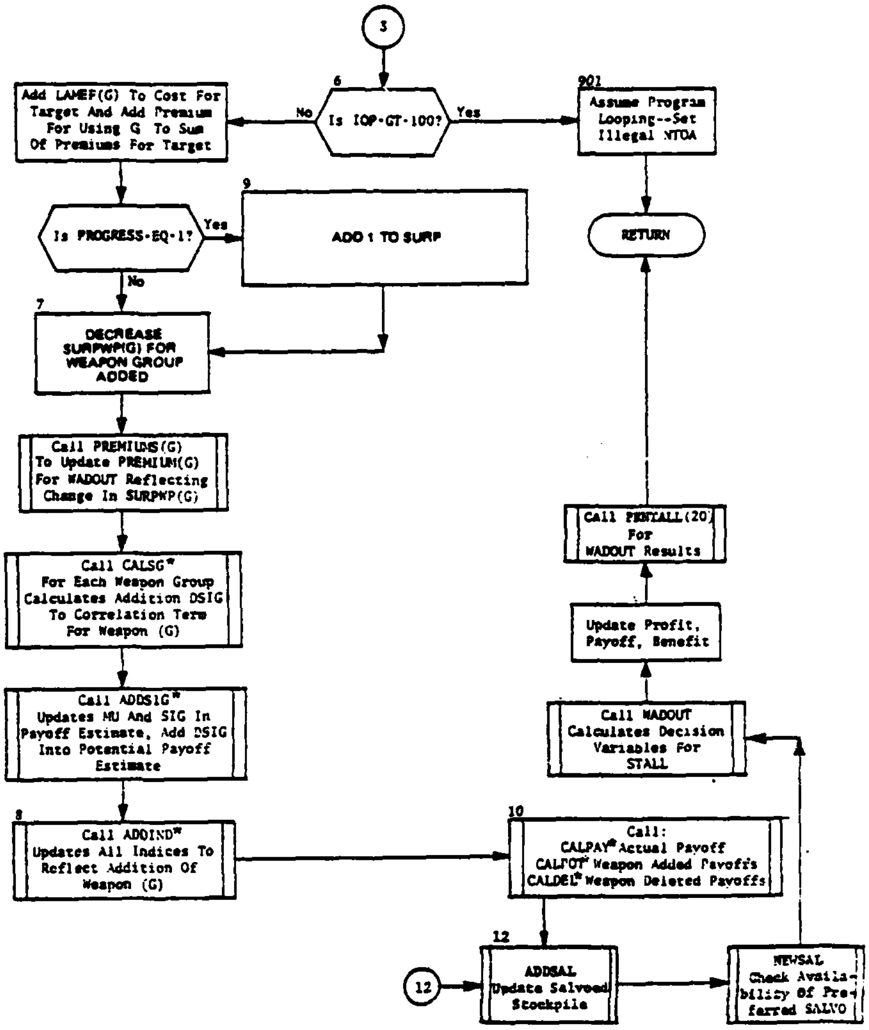
However, assuming that no such loop has occurred, the routine adds the Lagrange multiplier for the weapon added to the COST of the allocation and also updates SUMPREM, the sum of the premiums for the target. (This last variable SUMPREM, as well as the variables TBENEFIT and TPMX near statement 14, are computed only to provide a consistency check on the treatment of premiums. These variables are not essential to the operation of the program.) Notice that these variables and almost all other variables used by WAD such as VT, VTP, VTD, PAYOFF, PROFIT, etc. are computed (even for multiple targets) as if the program were dealing only with a single simple target. It is necessary to take target multiplicity into account only when dealing with variables which accumulate the total cost, total payoff, or total consumption of specific weapon groups over the whole target system.

The PREMIUMs used by the allocator, however, depend on just such a variable -- namely SURPWP(G), the available surplus (positive or negative) of unused weapons in each group G. Consequently, when a weapon G is added, the PREMIUM for that weapon group must be recalculated.

```
SURPWP(G) = SURPWP(G) -1
```

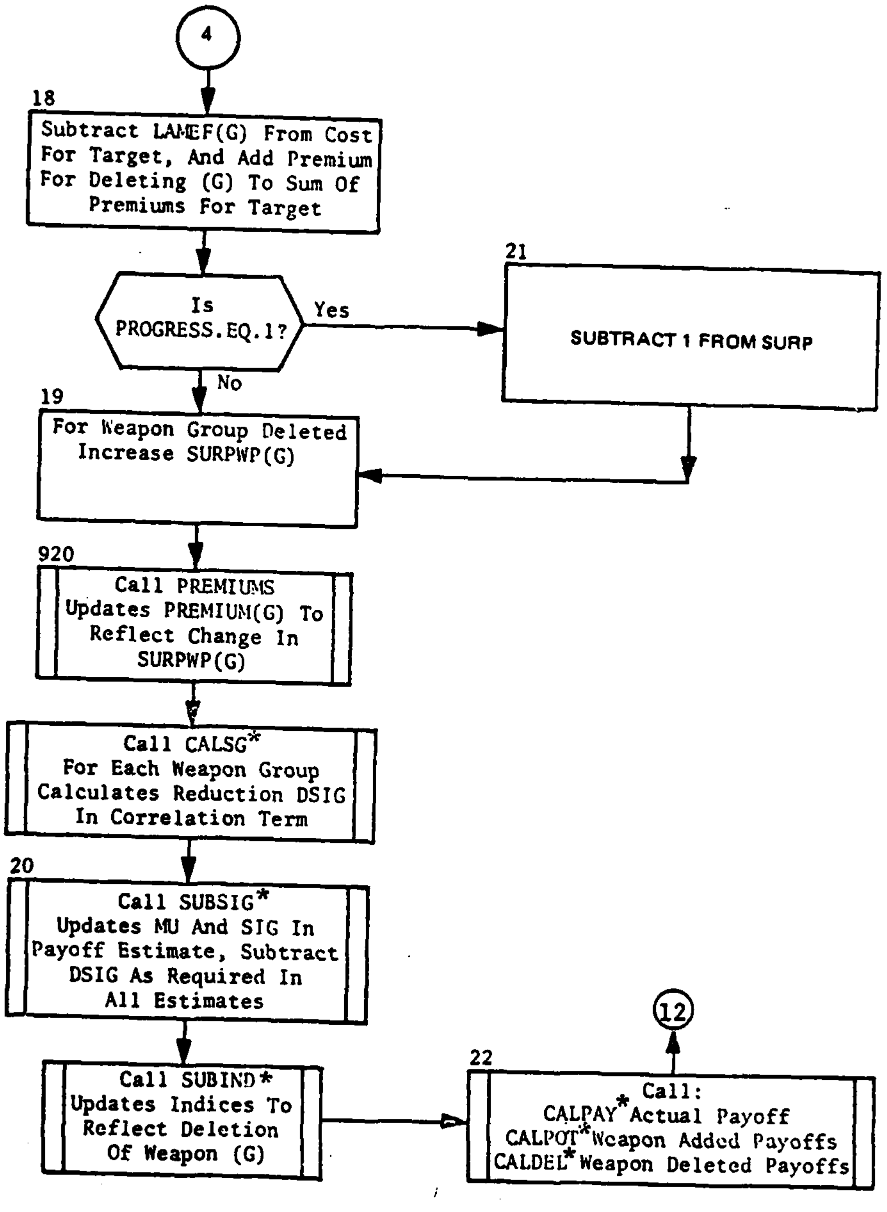
The program proceeds as usual to update SURPWP for the weapon now being added and PREMIUMS(G) is called as usual.

Both the add and delete weapon processes exit WAD through statement 12. At this statement, WAD calls: 1) ADDSAL to update the salvoed weapon stockpile; 2) NEWSAL to determine if the preferred salvo is still available, and 3) WADOUT to calculate the decision variables for STALL. After these calls, WAD updates the payoff, cost, profit and benefit variables.


3
6
Is IOP·GT·100?
No
Yes
Add LAMEF(G) To Cost For Target And Add Premium For Using G To Sum Of Premiums For Target
901
Assume Program Looping--Set Illegal NTOA
RETURN
Is PROGRESS·EQ·1?
Yes
No
9
ADD 1 TO SURF
7
DECREASE SURPWP(G) FOR WEAPON GROUP ADDED
Call PREMIUMS(G) To Update PREMIUM(G) For WADOUT Reflecting Change In SURPWP(G)
Call CALSG* For Each Weapon Group Calculates Addition DSIG To Correlation Term For Weapon (G)
Call ADDSIG* Updates MU And SIG In Payoff Estimate, Add DSIG Into Potential Payoff Estimate
8
Call ADDIND* Updates All Indices To Reflect Addition Of Weapon (G)
10
Call:
CALPAY* Actual Payoff
CALPOT* Weapon Added Payoffs
CALDEL* Weapon Deleted Payoffs
12
12
ADDSAL Update Salvoed Stockpile
NEWSAL Check Availability Of Preferred SALVO
Call WADOUT Calculates Decision Variables For STALL
Update Profit, Payoff, Benefit
Call PENTALL(20) For WADOUT Results

*Local subroutines

Figure 56. Part III: Add Weapon Control


4
18
Subtract LAMEF(G) From Cost For Target, And Add Premium For Deleting (G) To Sum Of Premiums For Target
Is PROGRESS.EQ.1?
Yes
No
21
SUBTRACT 1 FROM SURP
19
For Weapon Group Deleted Increase SURPWP(G)
920
Call PREMIUMS Updates PREMIUM(G) To Reflect Change In SURPWP(G)
Call CALSG* For Each Weapon Group Calculates Reduction DSIG In Correlation Term
20
Call SUBSIG* Updates MU And SIG In Payoff Estimate, Subtract DSIG As Required In All Estimates
Call SUBIND* Updates Indices To Reflect Deletion Of Weapon (G)
22
Call:
CALPAY* Actual Payoff
CALPOT* Weapon Added Payoffs
CALDEL* Weapon Deleted Payoffs
12

*Local subroutine

Figure 56. Part IV: Delete Weapon Control

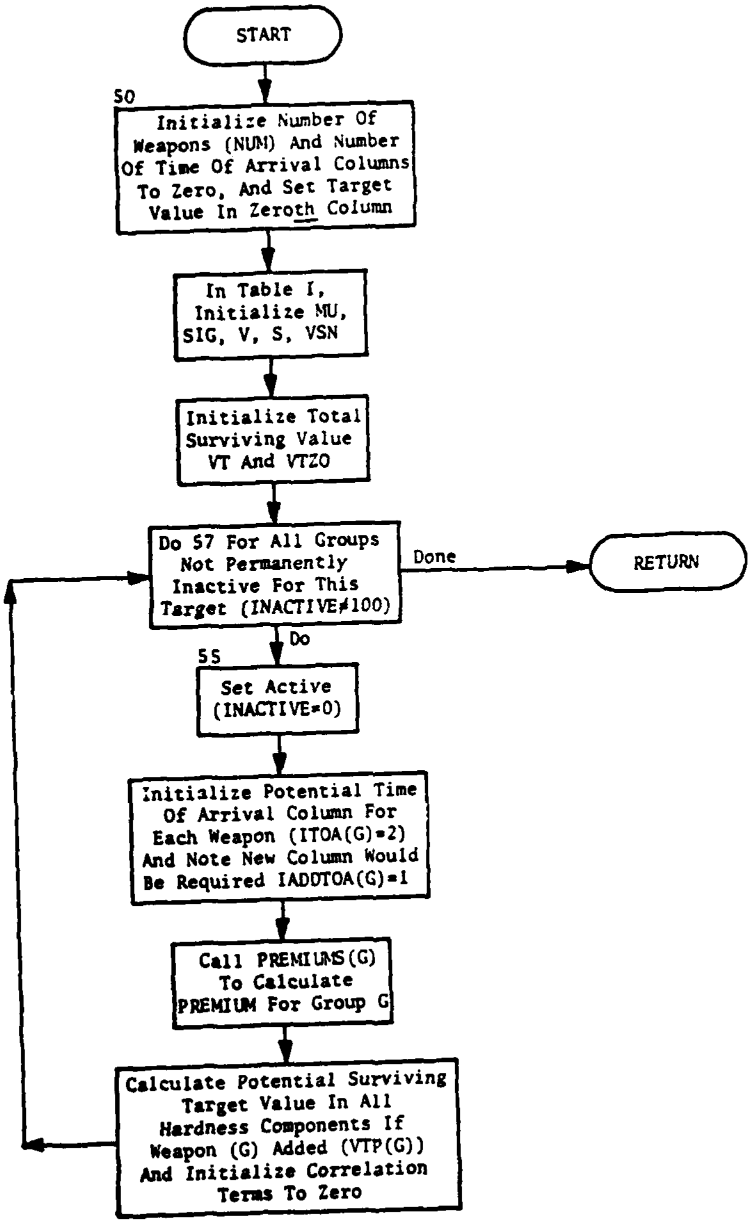

START
50
Initialize Number Of Weapons (NUM) And Number Of Time Of Arrival Columns To Zero, And Set Target Value In Zeroth Column
In Table I, Initialize MU, SIG, V, S, VSN
Initialize Total Surviving Value VT And VTZO
Do 57 For All Groups Not Permanently Inactive For This Target (INACTIVE≠100)
Done
RETURN
Do
55
Set Active (INACTIVE=0)
Initialize Potential Time Of Arrival Column For Each Weapon (ITOA(G)=2) And Note New Column Would Be Required IADDTOA(G)=1
Call PREMIUMS(G) To Calculate PREMIUM For Group G
Calculate Potential Surviving Target Value In All Hardness Components If Weapon (G) Added (VTP(G)) And Initialize Correlation Terms To Zero

Figure 56. Part V: Local Subroutine INITIALIZE

3.11.9 Subroutine WADOUT

PURPOSE: This routine summarizes decision alternatives for STALL by combining payoff data produced by WAD with the weapon cost data (Lagrange multipliers minus premiums). It also contributes to the efficiency of WAD by making inappropriate weapons inactive.

ENTRY POINTS: WADOUT

FORMAL PARAMETERS: None

COMMON BLOCKS: C30, C33, CONTRO, DYNAMI, PREMS, SALVO, WADFIN, WADOTX, WADWPN, WPFIX

SUBROUTINES CALLED: PRNTALL

CALLED BY: WAD

Method:

The output data produced by WADOUT consist of the following parameters for STALL that are recorded in /WADOTX/:

PPMX and IPPMX - the maximum potential increase in effective profit for any single weapon; and the index G to that weapon group, respectively.

PVRMX and IPVRMX - the maximum effective efficiency of any potential weapon; and the index G to that weapon group, respectively.

DPMN and IDPMN - the minimum marginal effective profit for any weapon now assigned; and the index NW to that weapon in the list of weapons assigned, respectively.

It also produces the array INACTIVE(G) in /WADWPN/ which is used by WAD to determine which weapons groups need not be processed.

The input data from WAD consist of:

VT - the surviving target value in /C33/

VTD(N) - the potential weapon deleted surviving target value (also in /DYNAMI/ (equivalenced to RVAL)

VTP(G) - the potential weapon-added surviving target value in /WADFIN/.

The input data on weapon costs consist of LAMEF(G), PREMIUM(G), and DPREMIUM(G).

WADOUT also initializes VTMAX and VTMIN of /WADOTX/, and MAXCOST which reflect the MINKILL, MAXKILL specifications for each target.

The quantities ALPHA and VTEF of /WADOTX/, are essentially local variables for WADOUT. They are included in this common block for use by PRNTNOW, and in the case of ALPHA, to allow WAD to reinitialize it to 1.0 for each new target.

The flow diagram, figure 57, is in three parts. In part I, the user-input parameter IMATCH is used to determine the method of computing MINKILL and MAXKILL. If IMATCH is 0, then the damage calculations used to determine residual target value for purposes of MINKILL and MAXKILL. use time dependence of target value. If IMATCH is nonzero, then MINKILL and MAXKILL are computed relative to the original target value. In addition, if IMATCH is 100, then the routine prints the WADOUT variables, VTO, VT, VTZO (original target value), FLGMN, FLGMX, SVTMIN, SVTMAX, VTMIN, VTMAX, and ALPHA.

The Do loop ending at statement 18 is used to flag all groups with fixed weapons on this target as active. This prevents their being removed during processing and maintains the validity of the damage calculations.

In Part II, the processing begins by evaluating the actual effective surviving target value VTEF. It then scans all weapons currently assigned to calculate the output quantities DPMN and IDPMN. If any weapon now on target fails to destroy a fraction of the original value greater than MINDAMAG, the weapon is flagged for immediate removal (statement 15). At the same time, the groups already assigned are flagged with INACTIVE = -100 to eliminate any possibility that they would be erroneously set inactive. (WADOUT never exits with INACTIVE set negative. A weapon group flagged with a -100 is always reset to INACTIVE = 0 before the routine exits.) (Do 14 loop)

Basically, the processing in part III is concerned with scanning all potential weapons to calculate the output quantities PPMX, IPPMX, PVRMX, and IPVRMX. Any weapon which would fail to destroy a fraction of the original value greater than MINDAMAG will be ignored in these calculations. Thus, it could never be allocated to the target.

In addition, if a weapon is a salvoed missile but no salvo is available (i.e., MYSAL(G) less than or equal to zero), or if the group has no weapons, then it is not considered as a potential weapon.

At the same time, however, the values for the array INACTIVE(G) are established. The INACTIVE array for each target is permanently stored on the WPNTGT files, as it was originally computed by GETDTA, with only two values -- zero for weapons in range of the target, and 100 for

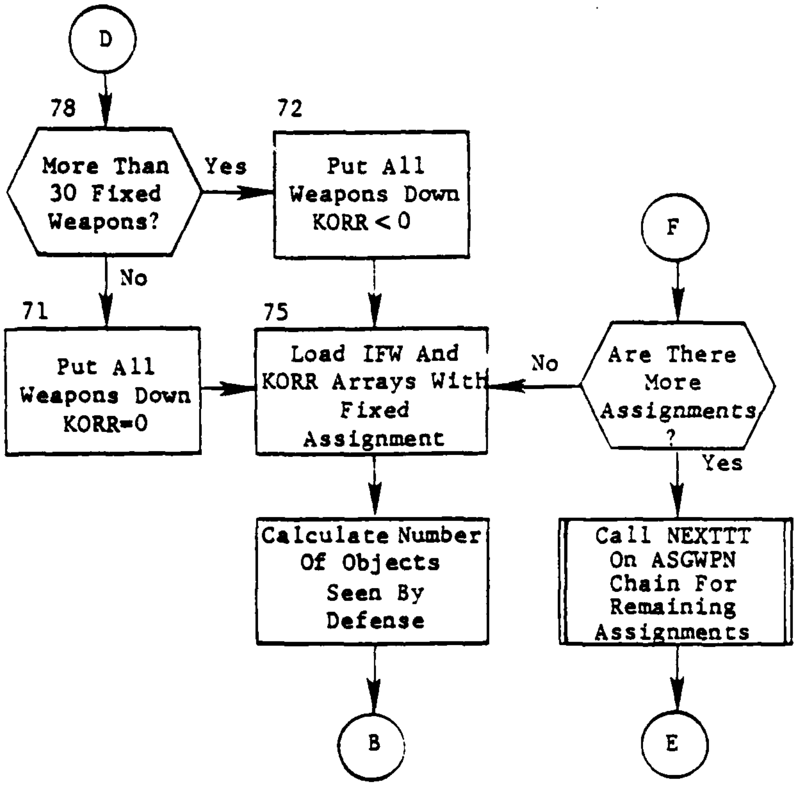

D
78
More Than 30 Fixed Weapons?
Yes
72
Put All Weapons Down KORR < 0
No
71
Put All Weapons Down KORR=0
75
Load IFW And KORR Arrays With Fixed Assignment
F
Are There More Assignments?
No
Yes
Calculate Number Of Objects Seen By Defense
Call NEXTTT On ASGWPN Chain For Remaining Assignments
B
E

Figure 58. Part II: (Part 2 of 2)

3.12.1 Subroutine PRNTOD

PURPOSE:	To produce optional prints for overlay DEFAL. (Options 27 and 28)
ENTRY POINTS:	PRNTOD
FORMAL PARAMETERS:	IOPT - Print option number
COMMON BLOCKS:	C30, DEFCOM, DEFRES
SUBROUTINES CALLED:	None
CALLED BY:	PRNTNOW

Method:

The formal parameter IOPT determines whether option 27 or 28 appears. The result of the option appears in the Users Manual, UM 9-77, Volume III.

Subroutine PRNTOD is illustrated in figure 59.

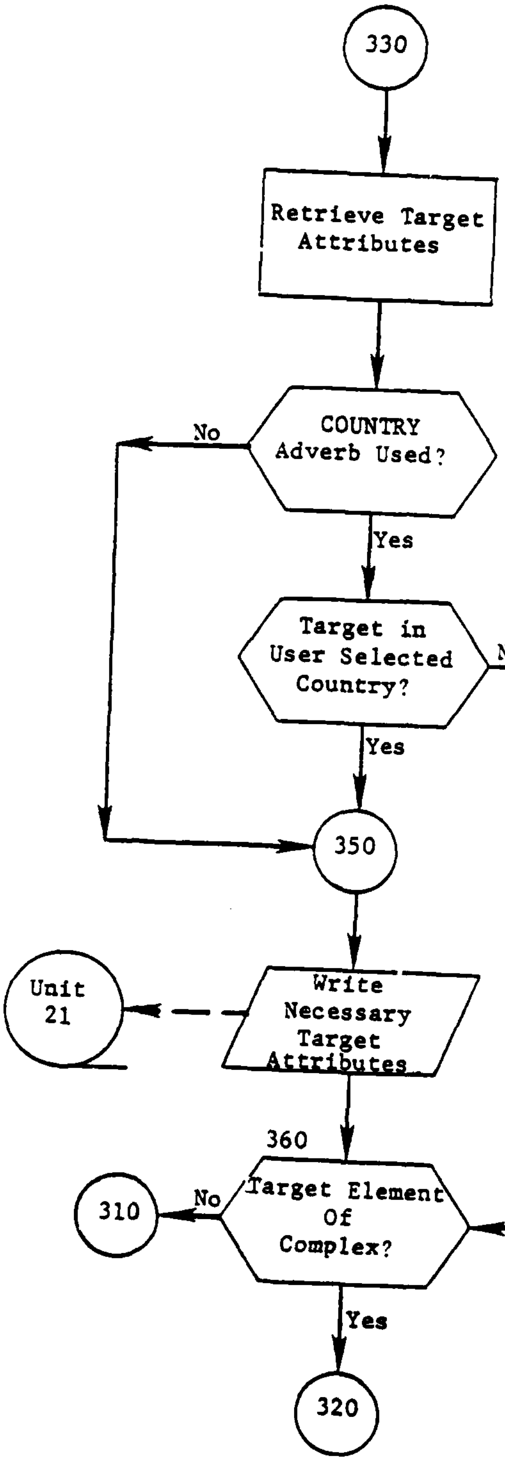

330
Retrieve Target Attributes
COUNTRY Adverb Used?
No
Yes
Target in User Selected Country?
No
Yes
350
Write Necessary Target Attributes
Unit 21
360
Target Element Of Complex?
No
310
Yes
320

Figure 63. (Part 4 of 7)

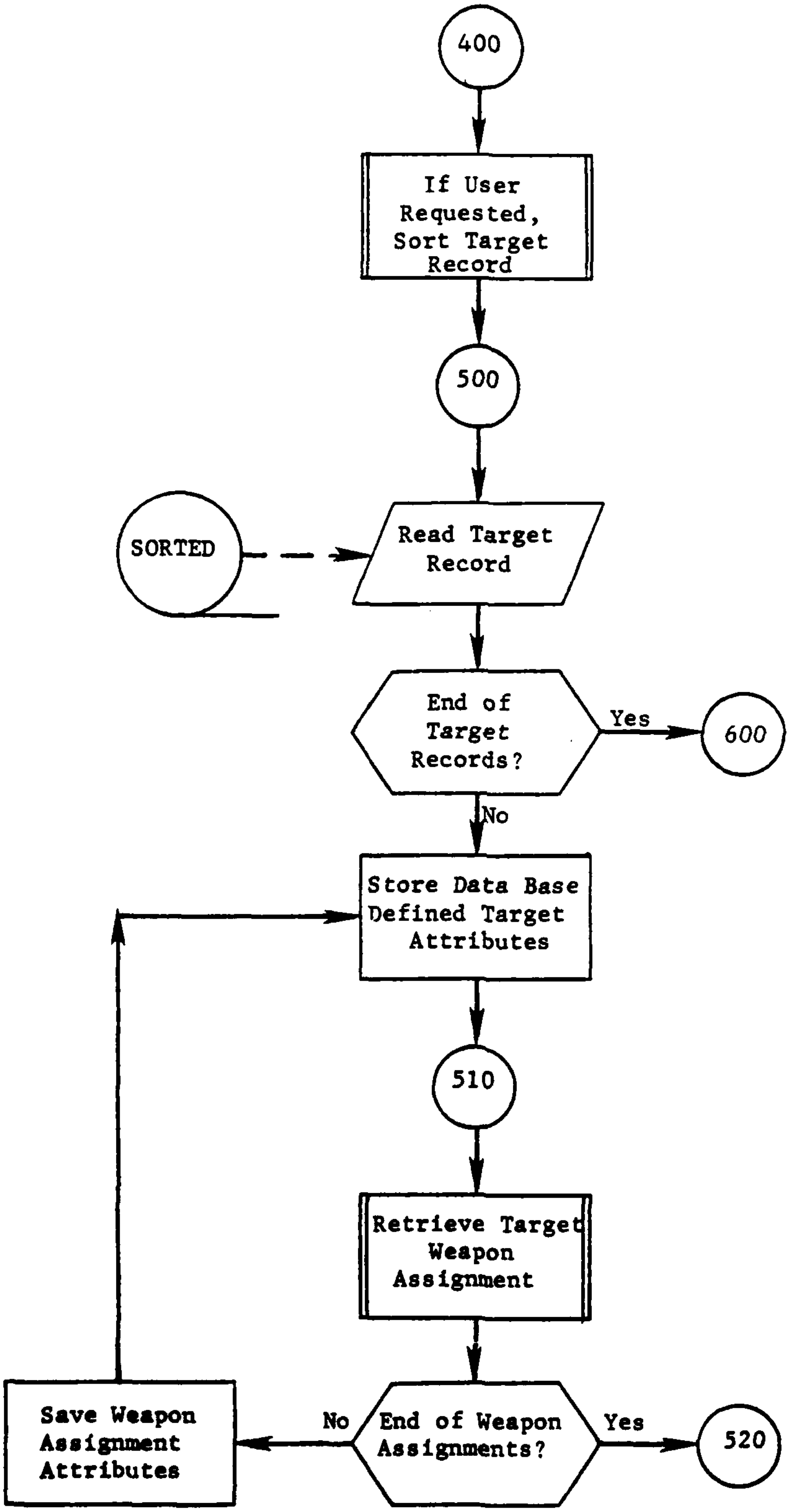

400
If User Requested, Sort Target Record
500
SORTED
Read Target Record
End of Target Records?
Yes
600
No
Store Data Base Defined Target Attributes
510
Retrieve Target Weapon Assignment
End of Weapon Assignments?
No
Yes
Save Weapon Assignment Attributes
520

Figure 63. (Part 5 of 7)

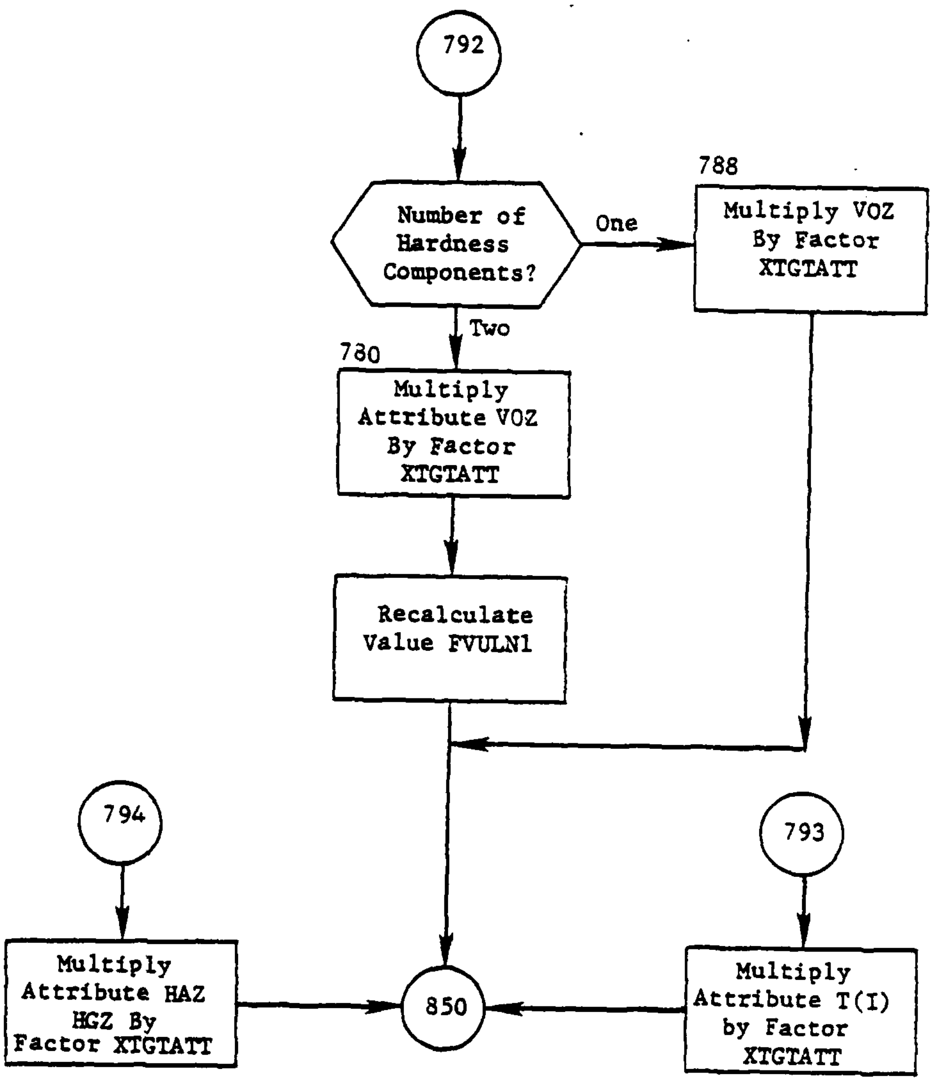

792
Number of Hardness Components?
One
788
Multiply VOZ By Factor XTGTATT
Two
780
Multiply Attribute VOZ By Factor XTGTATT
Recalculate Value FVULN1
794
793
Multiply Attribute HAZ HGZ By Factor XTGTATT
850
Multiply Attribute T(I) by Factor XTGTATT

Figure 66. (Part 2 of 4)

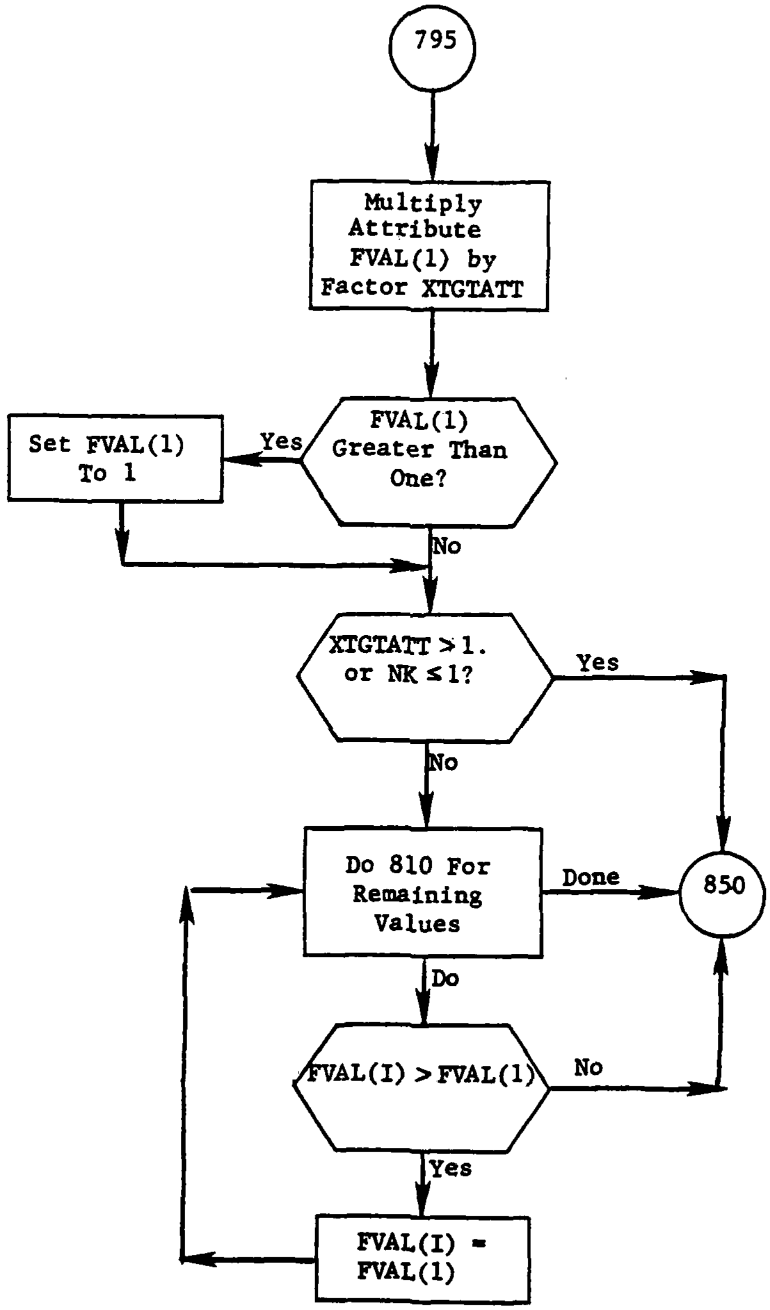


Figure 66. (Part 3 of 4)

SECTION 5. MODULE ALOCOUT

5.1 Purpose

Module ALOCOUT is responsible for selecting optimum DGZs (desired ground zeros), also called weapon aim points, for weapons allocated to target complexes. ALOCOUT also resorts weapon assignments at the group level for use within the Sortie Generation subsystem. If executed as part of mini-allocator process (DATAMAKE), produces a STRIKE format tape rather than resorting weapon assignments. (For format of STRIKE tape see CSM MM 9-77 Volume IV).

Module ALOC specifies weapon groups assigned to targets together with associated targeting data. ALOCOUT extracts data from these records and computes any aiming offsets required by the plan. For simple targets, no calculations are performed. In the case of complex targets which can have several elements at slightly different coordinates, ALOCOUT employs subroutine DGZ (desired ground zero selector) to select optimum aim points within the target complex.

5.2 Input

ALOCOUT operates after module ALOC assigned weapons to targets. These records (ASSIGN) in addition to the supporting data base structure must be defined for proper execution.

5.3 Output

No new data base records are created during the execution of ALOCOUT. However, the weapon assignment records (called ASSIGN) are modified in two ways. First, for assignments to complex targets or for assignments to cities with nonzero radius, offsets as determined within the module are included within the ASSIGN record. Second, the assignment records at the weapon group level are resorted for use within the Sortie Generation subsystem. For missile groups, the sort is based on decreasing values of attribute RVAL. For bomber groups, the order is based on penetration corridor index and within the corridor sorted based on attribute RVAL. The penetration corridor that contains the largest number of strikes appears first within the sort, followed by the penetration corridor of the next largest number of strikes and so on.

5.4 Concept of Operation

ALOCOUT (that is, subroutine ENTMOD) operates with three overlay links. The first overlay reads the target list (TARNUM) passes controls to subroutine PROCCOMP for offset calculations when applicable and finally supplies optional prints. After all targets have been processed controls passes to the second overlay which consists entirely of subroutine SUMPRN which reorders strikes at the weapon group level and, if requested, produces prints concerning the individual assignments. If part of mini-allocator, the second overlay is skipped and the third overlay is executed.

5.5 Identification of Subroutine Functions

5.5.1 Subroutine PROCCOMP. This subroutine controls the bulk of processing for offset determination. It is executed by subroutine ENTMOD only for those individual targets that require offset calculations. After offsets have been determined the assignment record (ASSIGN) is updated to include the values. Then, PROCCOMP returns to subroutine ENTMOD for acquisition of the next target and the associated strikes.

5.5.2 Subroutine SUMPRN. This subroutine constitutes the entire second overlay of ALOCOUT. Its purpose consists of resorting the weapon strikes at the group level and providing optional prints.

5.5.3 Subroutine MINIOUT. This subroutine controls the third overlay and produces a STRIKE tape.

5.6 Common Block Definition

Common blocks used by EVALALOC are outlined in table 11. Common blocks that communicate with the COP are given in appendix A of Program Maintenance Manual, Volume I.

Table 11. ALOCOUT Common Blocks
(Part 1 of 2)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
CITY	ICITY	Set to nonzero for targets with attribute RADIUS not equal to zero
C1	XO(J), YO(J)	Coordinates of target element J
	VI(J)	Initial target element values
	RADL(J)	Lethal radius of target element J
	VTOA(J,I)	Value of target element J immediately following arrival of weapon I
	S(J,I)	Survival probability of target element J relative to weapon I
	VEFF(J,I)	Effective value of target element J relative to weapon I
	X(I),Y(I)	Offset coordinates of weapon I
	PDEL(I)	Probability of delivery of weapon I
	ERDEL(I)	Error in delivery of weapon I
	YDSCL(I)	Scaled yield for weapon I
	VESC(I)	Intermediate computational value used in subroutine VAL for determination of total escaping target value
	NI	Number of weapons for complex
	NJ	Number of target elements for complex
GAMETIME	KDAY	Day set for STRIKE times (set to 1)
	KMON	Month set for STRIKE times (set to 1)
	KYEAR	Year set for STRIKE times (set to 1)
	HHR	Hour set for STRIKE times (set to 10)
GRPY	GROUPY(250)	Logical switch indicates if group allocated

Table 11. (Part 2 of 2)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
IONPRT	IPINDAT	User supplied print frequency for print option 1
	PRINCE(9)	Set TRUE if user requested option
ISKIPDGZ	ISKIPDGZ	Use indicator for DGZ. Normally it is 0. Compress resets it to 1 if more than 50 calls to it are made to reduce the number of target elements for a complex target; DGZ is not used again for the target in this case
JAZ	F(400,26)	Holding arrays for sort purposes
LOCFIN	LOCFIN	Starting location into IRSET's arrays for adverb FINDMIN instructions
STRIKE	TOA(I)	Weapon time of arrival to target
	IREFSTRK(I)	Reference code of weapon strike
	N	Number of strikes
WPGT	YDMIN	Minimum allowable weapon group yield
	IGRP	Group number

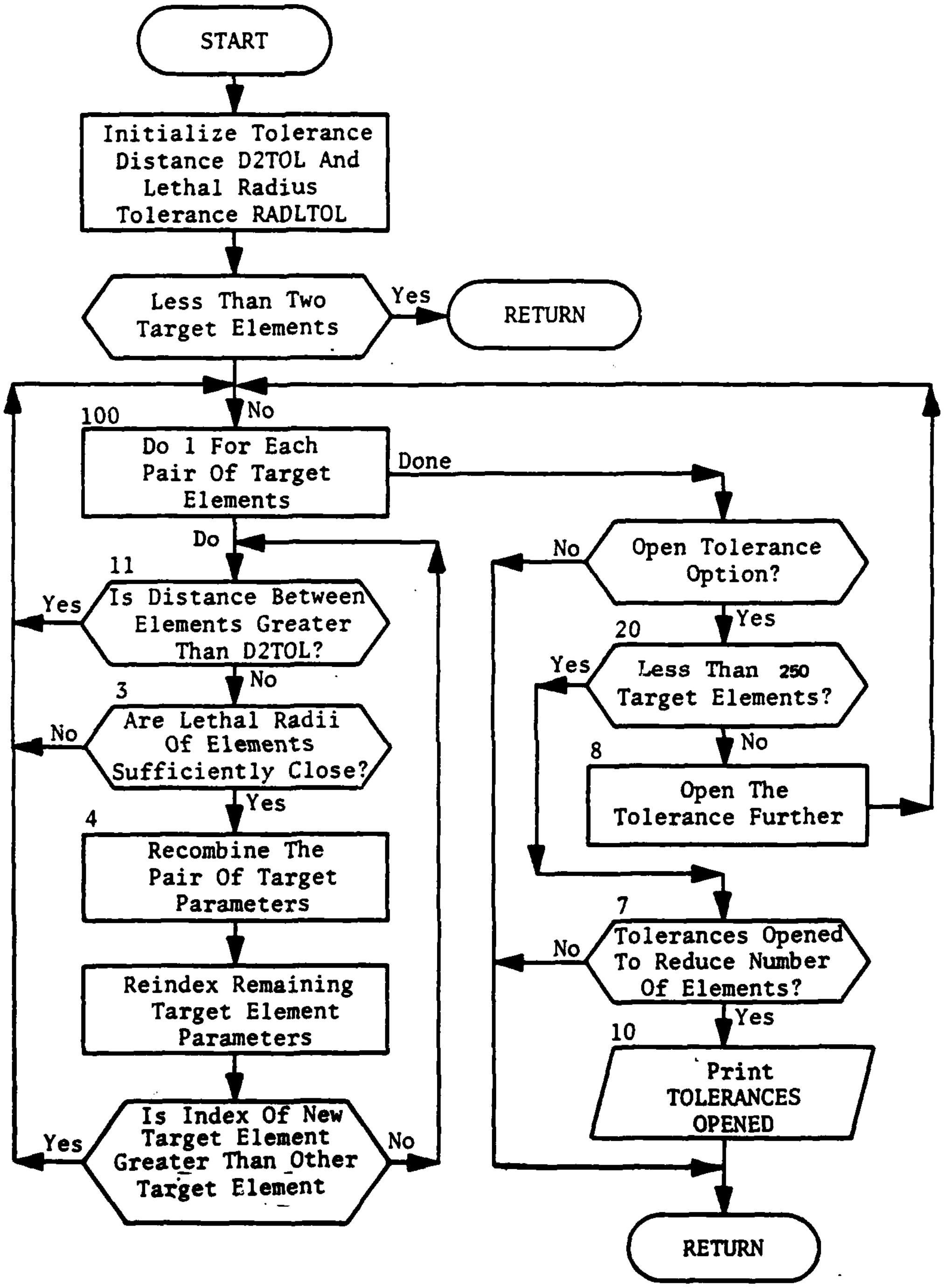

START
Initialize Tolerance Distance D2TOL And Lethal Radius Tolerance RADLTOL
Less Than Two Target Elements
Yes
RETURN
No
100
Do 1 For Each Pair Of Target Elements
Done
Do
11
Is Distance Between Elements Greater Than D2TOL?
Yes
No
3
Are Lethal Radii Of Elements Sufficiently Close?
No
Yes
4
Recombine The Pair Of Target Parameters
Reindex Remaining Target Element Parameters
Is Index Of New Target Element Greater Than Other Target Element
Yes
No
Open Tolerance Option?
No
Yes
20
Less Than 250 Target Elements?
Yes
No
8
Open The Tolerance Further
7
Tolerances Opened To Reduce Number Of Elements?
No
Yes
10
Print TOLERANCES OPENED
RETURN

Figure 69. Subroutine COMPRESS

5.7.2 Function CUMINV

PURPOSE:	To determine the value X such that Z is the probability that $x \leq X$.
ENTRY POINTS:	CUMINV
FORMAL PARAMETERS:	Z - The probability that $x \leq X$
COMMON BLOCKS:	None
SUBROUTINES CALLED:	None
CALLED BY:	PROCCOMP

Method:

Function CUMINV is illustrated in figure 70. By definition,

$$Z = P[x \leq X] = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{X} e^{-\frac{t^2}{2}} \cdot dt \text{ for } 0 < Z < 1$$

CUMINV uses the following approximation X^1 for X:

$$X^1 = \pm \left[V - \left(\frac{A_1 + A_2 \cdot V + A_3 \cdot V^2}{1 + B_1 \cdot V + B_2 \cdot V^2 + B_3 \cdot V^3} \right) \right]$$

where

$$V = \sqrt{\ln (1/Q^2)}, \quad Q = Z \quad \text{or} \quad 1 - Z \quad \text{such that } 0 \leq Q \leq .5$$

and

$A_1 = 2.515517$	$B_1 = 1.432788$
$A_2 = .802853$	$B_2 = .189269$
$A_3 = .010328$	$B_3 = .001308$

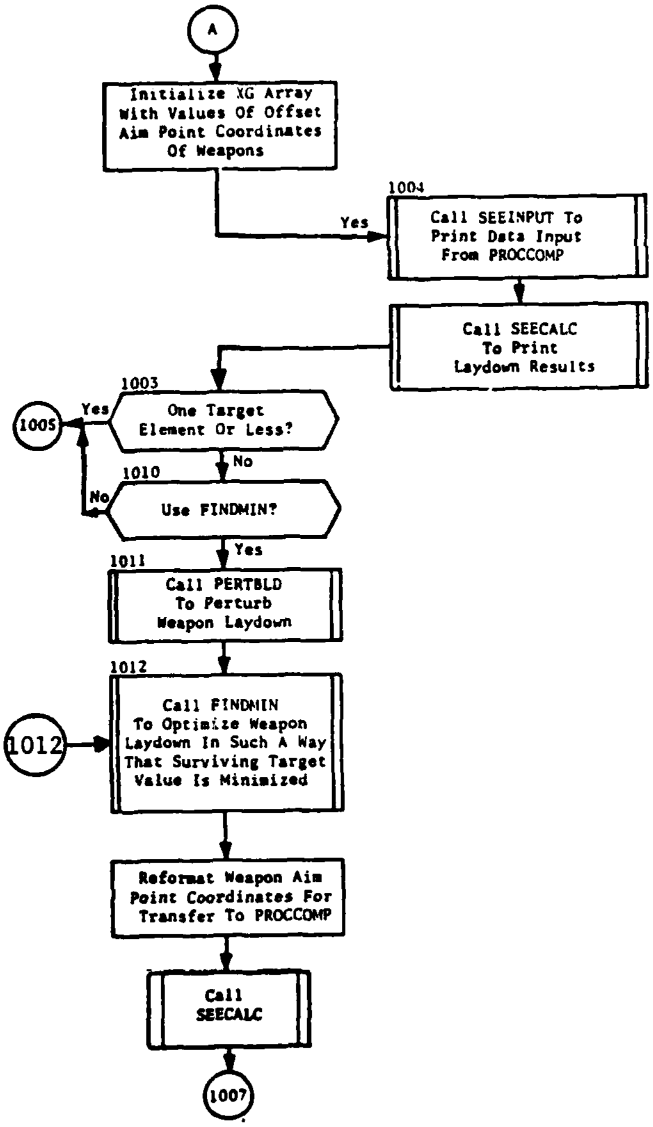

A
Initialize XG Array With Values Of Offset Aim Point Coordinates Of Weapons
Yes
1004
Call SEEINPUT To Print Data Input From PROCCOMP
Call SEECALC To Print Laydown Results
1003
One Target Element Or Less?
Yes
1005
No
1010
Use FINDMIN?
No
Yes
1011
Call PERTBLD To Perturb Weapon Laydown
1012
Call FINDMIN To Optimize Weapon Laydown In Such A Way That Surviving Target Value Is Minimized
1012
Reformat Weapon Aim Point Coordinates For Transfer To PROCCOMP
Call SEECALC
1007

Figure 72. (Part 2 of 3)

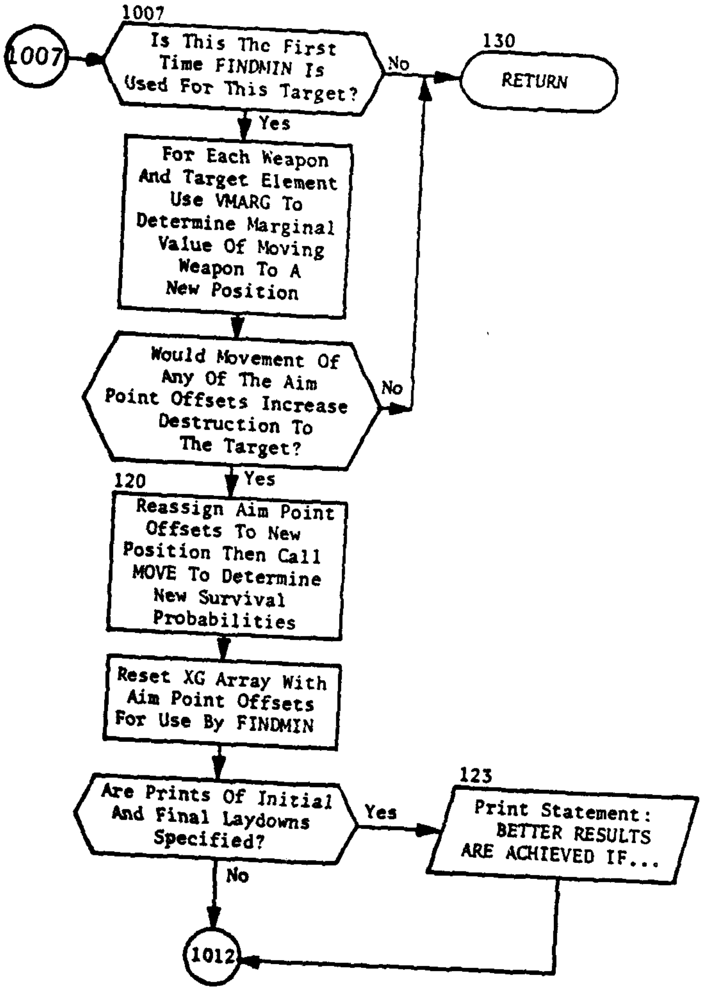

1007
1007
Is This The First Time FINDMIN Is Used For This Target?
No
130
RETURN
Yes
For Each Weapon And Target Element Use VMARG To Determine Marginal Value Of Moving Weapon To A New Position
Would Movement Of Any Of The Aim Point Offsets Increase Destruction To The Target?
No
Yes
120
Reassign Aim Point Offsets To New Position Then Call MOVE To Determine New Survival Probabilities
Reset XG Array With Aim Point Offsets For Use By FINDMIN
Are Prints Of Initial And Final Laydowns Specified?
Yes
123
Print Statement: BETTER RESULTS ARE ACHIEVED IF...
No
1012

Figure 72. (Part 3 of 3)

5.7.5 Subroutine FINDMIN

PURPOSE: This subroutine uses a steepest descent method to determine a local minimum of a function of several variables. An initial estimate of the minimum position is input, together with various tolerances. FINDMIN uses two auxiliary routines, F2BMIN and GRADF, which define the function to be minimized and its gradient, respectively. DGZ uses FINDMIN to find the DGZs for complex targets.

ENTRY POINTS: FINDMIN

FORMAL PARAMETERS:

XO - Initial guess at aim point offsets
N - Length of XO vector
IMAX - Maximum number of iteractions for FINDMIN
E1,E2 - Tolerances for the minimization
X - Best aim point offsets as determined by FINDMIN
F1 - Minimum value found for escaped target value
IFLAG - Print control flag
>0 : FINDMIN debug prints will be produced

SUBROUTINES CALLED: F2BMIN, GRADF, SEECALC

CALLED BY: DGZ

Method:

Given a function $F(X_1, X_2)$, the gradients $G_1 = \frac{\partial F}{\partial X_1}$, and $G_2 = \frac{\partial F}{\partial X_2}$, and an initial guess (XO_1, XO_2) at the aim point offsets, FINDMIN finds the local escaped target value F and its associated aim point offset coordinates (X_1, X_2). Each iteration consists of a function, F, and gradient, (G_1, G_2), evaluation followed by determination of the minimum function value along the line associated with the modified steepest descent direction. F is redetermined at each iteration and is defined in such a way that it converges after two iterations. FINDMIN uses two subroutines during its processing. The first, F2BMIN, defines the escaped target value function F in terms of aim point offsets X_1 and X_2. The second, GRADF, defines the gradient components G_1 and G_2.

Subroutine SEECALC is called after each iteration to print the results of the optimization.

Subroutine FINDMIN is illustrated in figure 74.

AD-A085 289 COMMAND AND CONTROL TECHNICAL CENTER WASHINGTON DC F/G 15/7
THE CCTC QUICK-REACTING GENERAL WAR GAMING SYSTEM (QUICK). PROG--ETC(U)
APR 80

UNCLASSIFIED NL

3 OF 3

AD
A085289

END
DATE
FILMED
7-80
DTIC

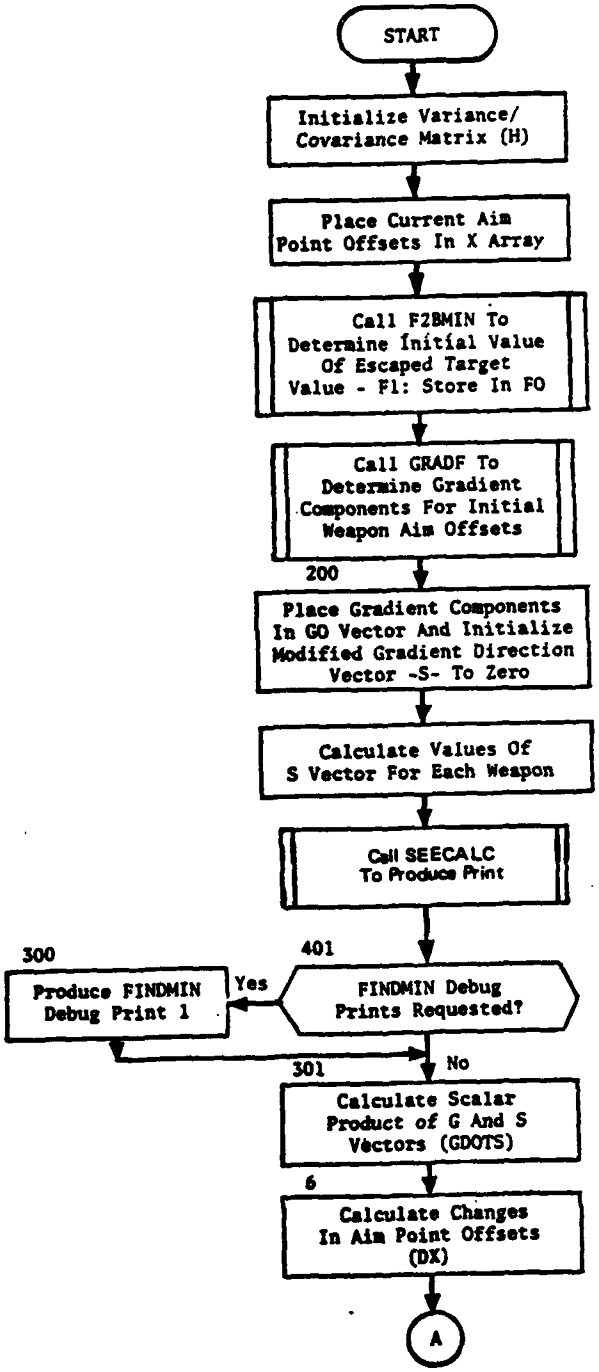

START
Initialize Variance/ Covariance Matrix (H)
Place Current Aim Point Offsets In X Array
Call F2BMIN To Determine Initial Value Of Escaped Target Value - F1: Store In FO
Call GRADF To Determine Gradient Components For Initial Weapon Aim Offsets
200
Place Gradient Components In GO Vector And Initialize Modified Gradient Direction Vector -S- To Zero
Calculate Values Of S Vector For Each Weapon
Call SEECALC To Produce Print
401
FINDMIN Debug Prints Requested?
Yes
300
Produce FINDMIN Debug Print 1
No
301
Calculate Scalar Product of G And S Vectors (GDOTS)
6
Calculate Changes In Aim Point Offsets (DX)
A

Figure 74. Subroutine FINDMIN
(Part 1 of 4)

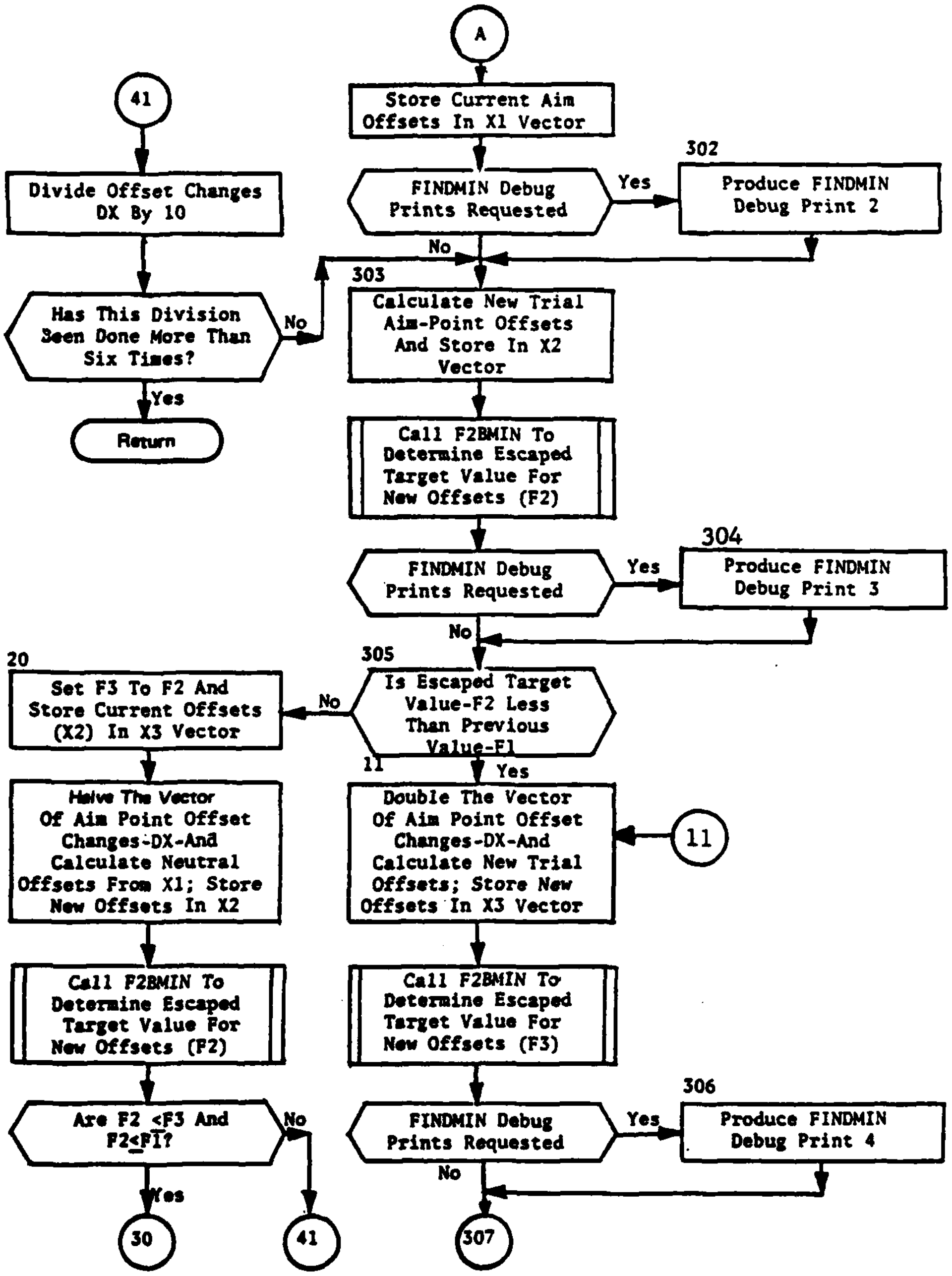

A
41
Store Current Aim Offsets In X1 Vector
302
FINDMIN Debug Prints Requested
Yes
Produce FINDMIN Debug Print 2
Divide Offset Changes DX By 10
No
303
Has This Division Been Done More Than Six Times?
No
Calculate New Trial Aim-Point Offsets And Store In X2 Vector
Yes
Return
Call F2BMIN To Determine Escaped Target Value For New Offsets (F2)
304
FINDMIN Debug Prints Requested
Yes
Produce FINDMIN Debug Print 3
No
20
305
Set F3 To F2 And Store Current Offsets (X2) In X3 Vector
No
Is Escaped Target Value-F2 Less Than Previous Value-F1
11
Yes
Halve The Vector Of Aim Point Offset Changes-DX-And Calculate Neutral Offsets From X1; Store New Offsets In X2
Double The Vector Of Aim Point Offset Changes-DX-And Calculate New Trial Offsets; Store New Offsets In X3 Vector
11
Call F2BMIN To Determine Escaped Target Value For New Offsets (F2)
Call F2BMIN To Determine Escaped Target Value For New Offsets (F3)
306
Are F2 <F3 And F2<F1?
No
FINDMIN Debug Prints Requested
Yes
Produce FINDMIN Debug Print 4
No
Yes
30
41
307

Figure 74. (Part 2 of 4)

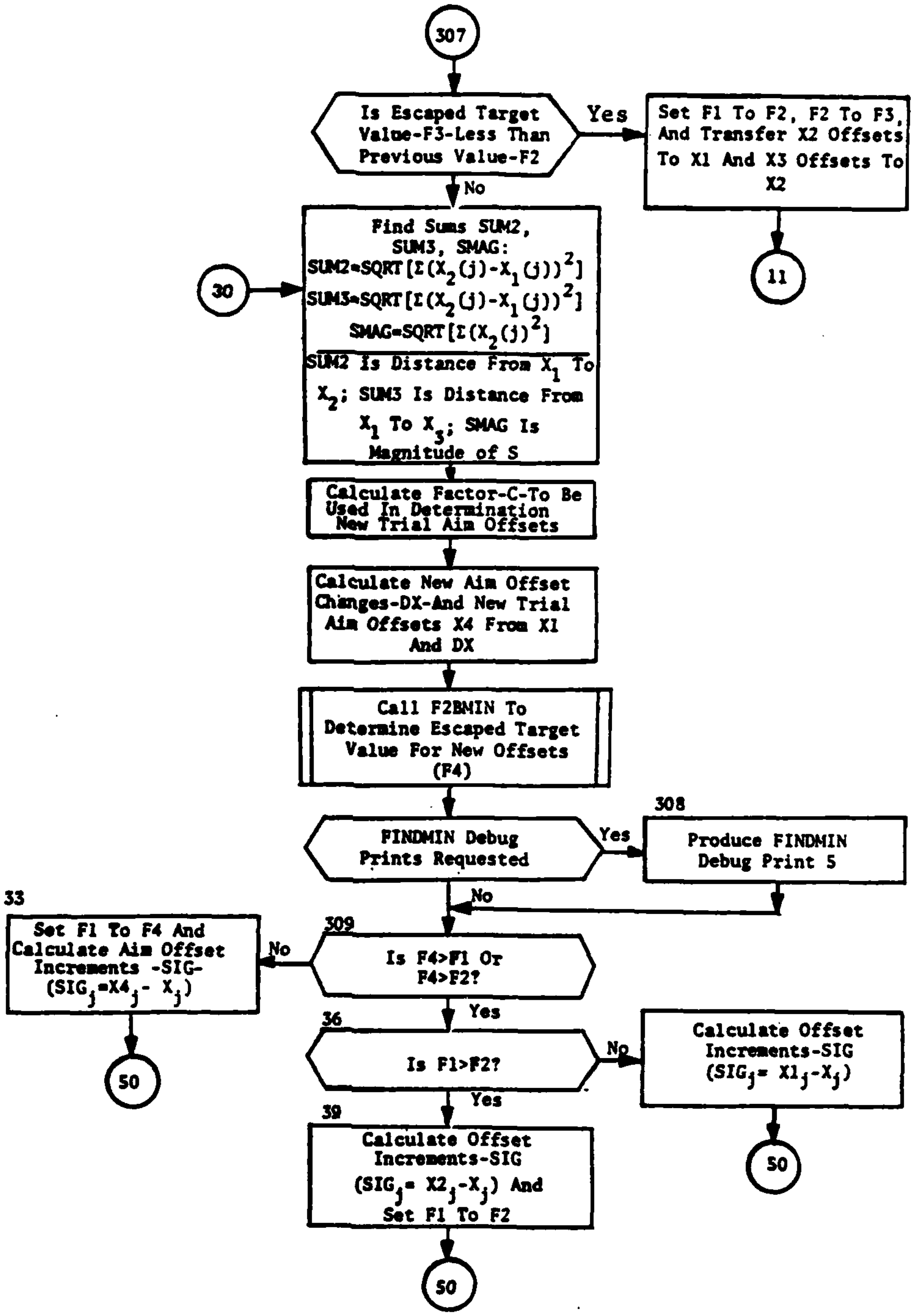

307
Is Escaped Target Value-F3-Less Than Previous Value-F2
Yes
Set F1 To F2, F2 To F3, And Transfer X2 Offsets To X1 And X3 Offsets To X2
11
No
30
Find Sums SUM2, SUM3, SMAG:
SUM2=SQRT$[\Sigma(X_2(j)-X_1(j))^2]$
SUM3=SQRT$[\Sigma(X_2(j)-X_1(j))^2]$
SMAG=SQRT$[\Sigma(X_2(j)^2]$
SUM2 Is Distance From X_1 To X_2; SUM3 Is Distance From X_1 To X_3; SMAG Is Magnitude of S
Calculate Factor-C-To Be Used In Determination New Trial Aim Offsets
Calculate New Aim Offset Changes-DX-And New Trial Aim Offsets X4 From X1 And DX
Call F2BMIN To Determine Escaped Target Value For New Offsets (F4)
308
FINDMIN Debug Prints Requested
Yes
Produce FINDMIN Debug Print 5
No
33
Set F1 To F4 And Calculate Aim Offset Increments -SIG- $(SIG_j=X4_j-X_j)$
No
309
Is F4>F1 Or F4>F2?
50
Yes
36
Is F1>F2?
No
Calculate Offset Increments-SIG $(SIG_j=X1_j-X_j)$
Yes
39
Calculate Offset Increments-SIG $(SIG_j=X2_j-X_j)$ And Set F1 To F2
50
50

Figure 74. (Part 3 of 4)

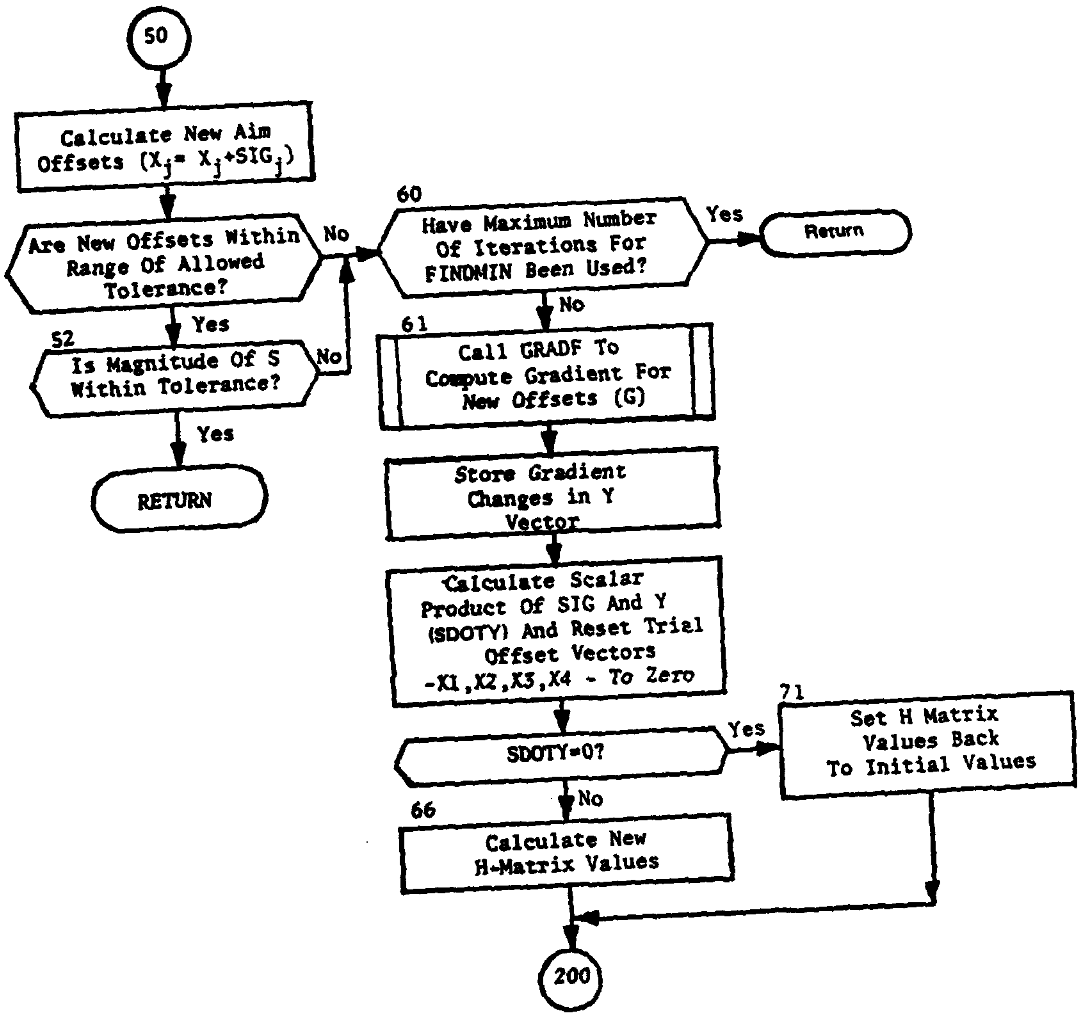

50
Calculate New Aim Offsets ($X_j = X_j+SIG_j$)
Are New Offsets Within Range Of Allowed Tolerance?
No
Yes
52
Is Magnitude Of S Within Tolerance?
No
Yes
RETURN
60
Have Maximum Number Of Iterations For FINDMIN Been Used?
Yes
Return
No
61
Call GRADF To Compute Gradient For New Offsets (G)
Store Gradient Changes in Y Vector
Calculate Scalar Product Of SIG And Y (SDOTY) And Reset Trial Offset Vectors -X1,X2,X3,X4 - To Zero
SDOTY=0?
Yes
71
Set H Matrix Values Back To Initial Values
No
66
Calculate New H-Matrix Values
200

Figure 74. (Part 4 of 4)

5.7.10 Subroutine PROCCOMP

PURPOSE:	To set up arrays in common block /C1/ for the complex target so that the subroutine DGZ can use the arrays during the selection of optimal aim point offsets for the weapons allocated to the target; and to modify target weapon assignments records for inclusion of the computed offsets.
ENTRY POINTS:	PROCCOMP
FORMAL PARAMETERS:	None
COMMON BLOCKS:	CITY, C1, C10, C30, ISKIPD, STRIKE, WPGT
SUBROUTINES CALLED:	COMPRESS, CUMINV, DGZ, DIRECT, ERGOT1, ERGOT2, HEAD, MODFY, NEXTTT, ORDER, REORDER, TIMEME, VALTAR
CALLED BY:	ENTMOD

Method:

When ENTMOD encounters a complex target, PROCCOMP is called in order to assemble data in a form that can be efficiently used for DGZ selection. Each target component of the complex generates a standardized "target element" in the working arrays used by subroutine DGZ (common /C1/). Targets with more than one hardness component generate more than one such target element, and targets with a specified target radius will generate several elements spread over the area of the target to represent a value over the area. For complexes, individual target elements are obtained by walking the data base chain called 'CMPTGT'.

If the number of target elements so generated exceeds the maximum program dimensions (290), subroutine COMPRESS is called to recombine target elements near each other having nearly the same lethal radius. In any case, for efficiency in DGZ, a call to COMPRESS is made just before calling DGZ. On return from DGZ, PROCCOMP modifies weapon assignment records (ASSIGN) for definition of the computed offsets.

Subroutine PROCCOMP is illustrated in figure 79.

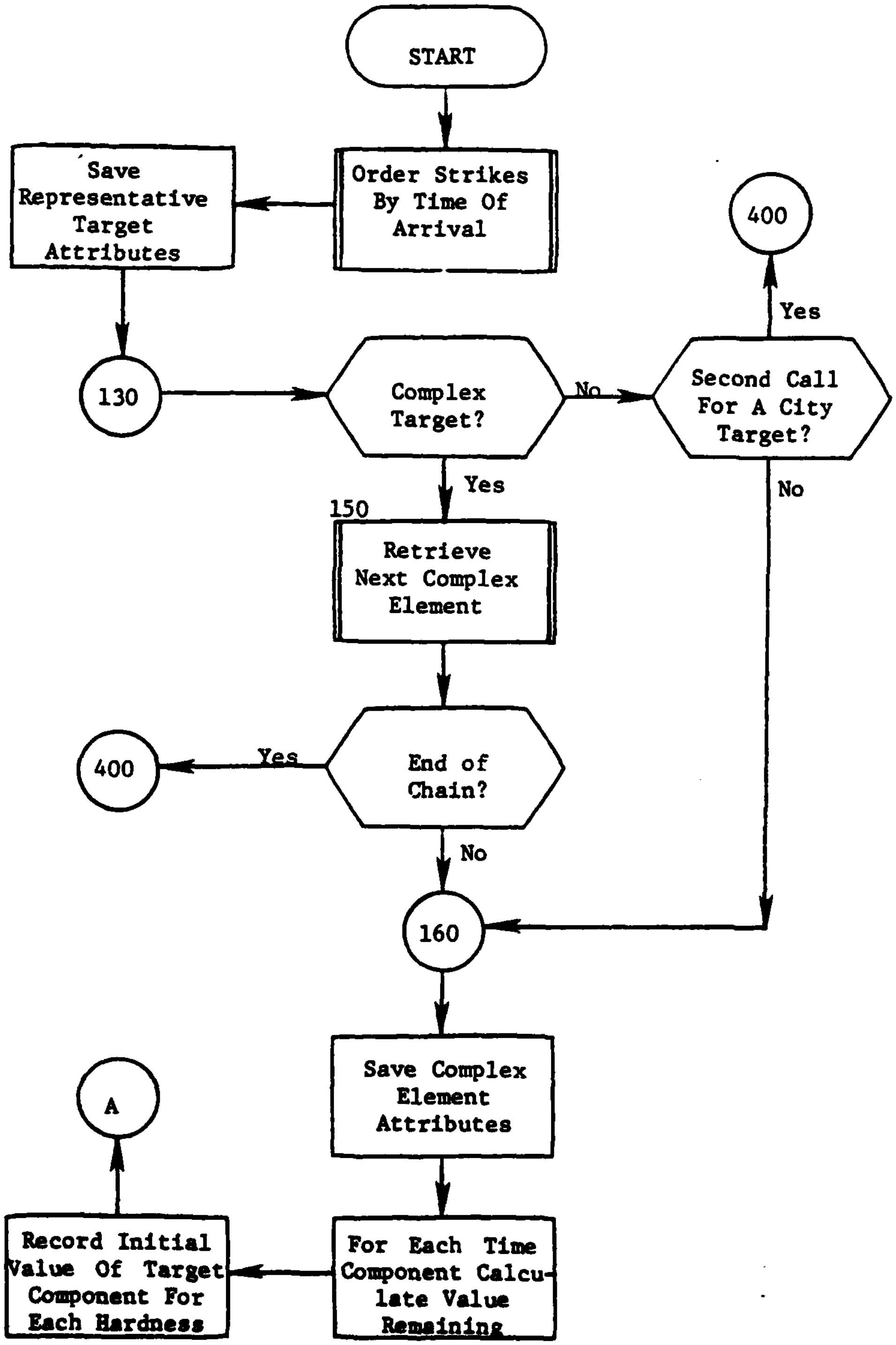

START
Order Strikes By Time Of Arrival
Save Representative Target Attributes
400
Yes
130
Complex Target?
No
Second Call For A City Target?
Yes
No
150
Retrieve Next Complex Element
400
Yes
End of Chain?
No
160
Save Complex Element Attributes
A
Record Initial Value Of Target Component For Each Hardness
For Each Time Component Calculate Value Remaining

Figure 79. Subroutine PROCCOMP (Part 1 of 4)

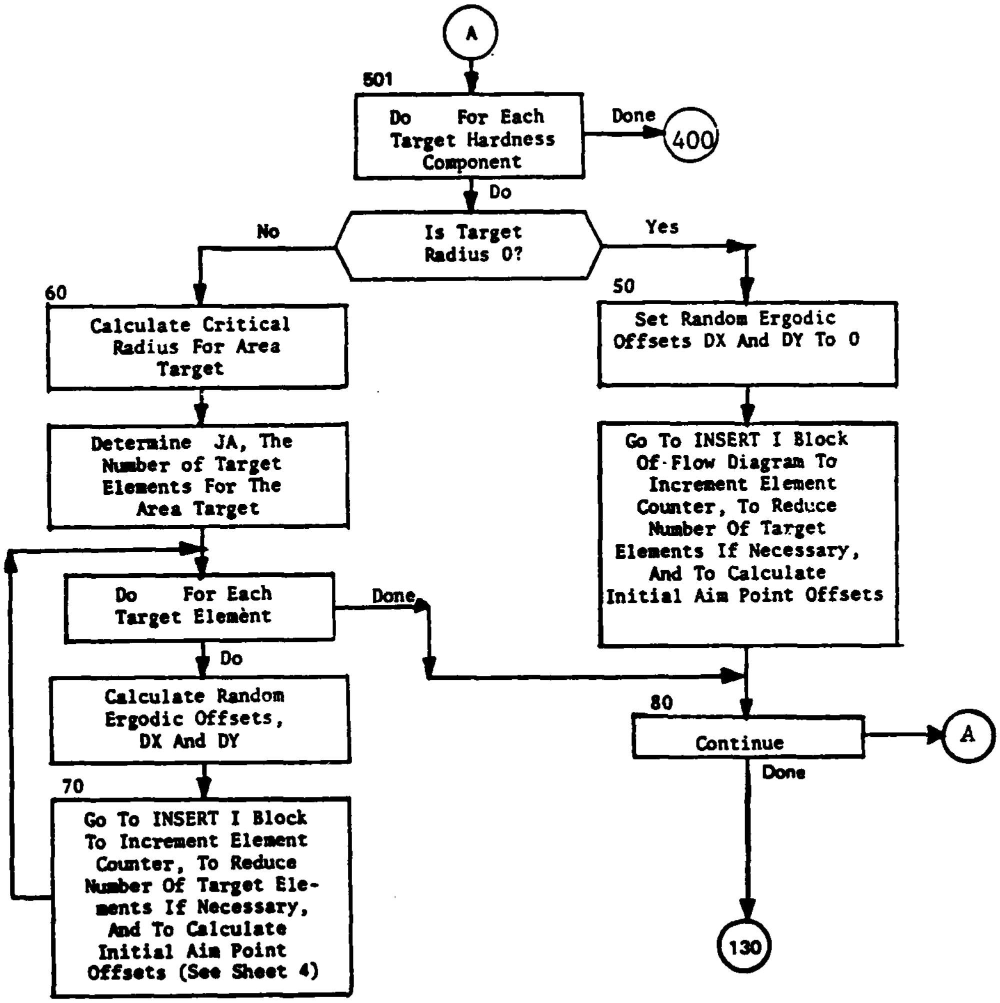

A
501
Do For Each Target Hardness Component
Done
400
Do
Is Target Radius 0?
No
Yes
60
Calculate Critical Radius For Area Target
50
Set Random Ergodic Offsets DX And DY To 0
Determine JA, The Number of Target Elements For The Area Target
Go To INSERT I Block Of-Flow Diagram To Increment Element Counter, To Reduce Number Of Target Elements If Necessary, And To Calculate Initial Aim Point Offsets
Do For Each Target Elemènt
Done
Do
Calculate Random Ergodic Offsets, DX And DY
80
Continue
A
Done
70
Go To INSERT I Block To Increment Element Counter, To Reduce Number Of Target Elements If Necessary, And To Calculate Initial Aim Point Offsets (See Sheet 4)
130

Figure 79. (Part 2 of 4)

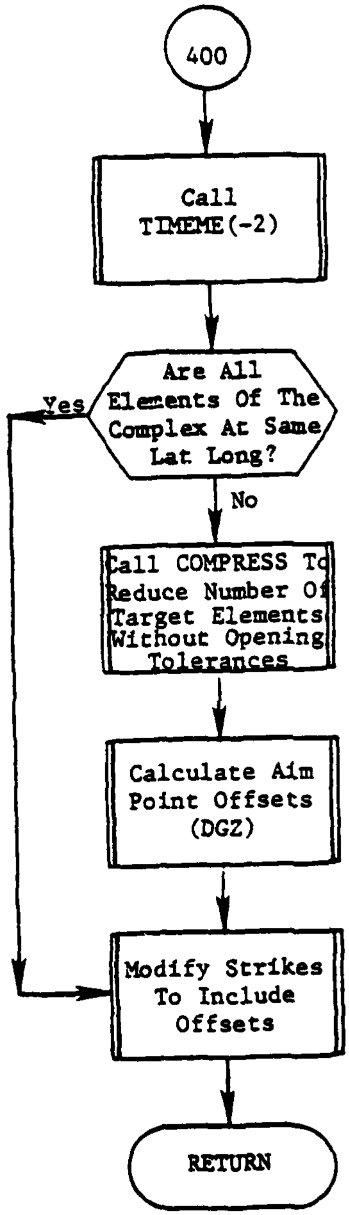

400
Call TIMEME(-2)
Are All Elements Of The Complex At Same Lat Long?
Yes
No
Call COMPRESS To Reduce Number Of Target Elements Without Opening Tolerances
Calculate Aim Point Offsets (DGZ)
Modify Strikes To Include Offsets
RETURN

Figure 79. (Part 3 of 4)

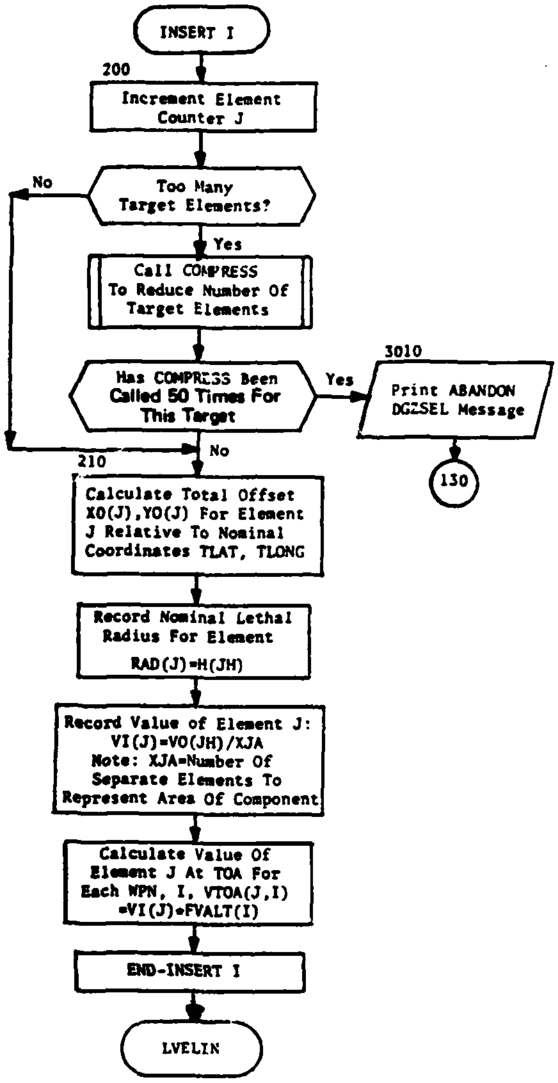

INSERT I
200
Increment Element Counter J
Too Many Target Elements?
No
Yes
Call COMPRESS To Reduce Number Of Target Elements
Has COMPRESS Been Called 50 Times For This Target
Yes
3010
Print ABANDON DGZSEL Message
130
No
210
Calculate Total Offset XO(J),YO(J) For Element J Relative To Nominal Coordinates TLAT, TLONG
Record Nominal Lethal Radius For Element RAD(J)=H(JH)
Record Value of Element J: VI(J)=VO(JH)/XJA Note: XJA=Number Of Separate Elements To Represent Area Of Component
Calculate Value Of Element J At TOA For Each WPN, I, VTOA(J,I) =VI(J)*FVALT(I)
END-INSERT I
LVELIN

Figure 79. (Part 4 of 4)

5.7.11 Subroutine SEECALC

PURPOSE: To print the computation values relevant to the selection of aim point offsets at various points within the DGZSEL subarea of program ALOCOUT.

ENTRY POINTS: SEECALC

FORMAL PARAMETERS:

VESCTOT	:	Total escaping target value
XX	:	Vector containing the aim point offset positions for the weapons

COMMON BLOCKS: C1, WAROUT

CALLED BY: DGZ, FINDMIN

Method:

When called by DGZ or FINDMIN subroutine SEECALC prints the title DGZSEL COMPUTATION VALUES and column headings. Then for each weapon allocated to the target, SEECALC prints the internal weapon number, the aim point offsets, and the survival of each target element relative to the weapon. At the end of the print for each target, the total escaped target value is printed.

Subroutine SEECALC is illustrated in figure 80.

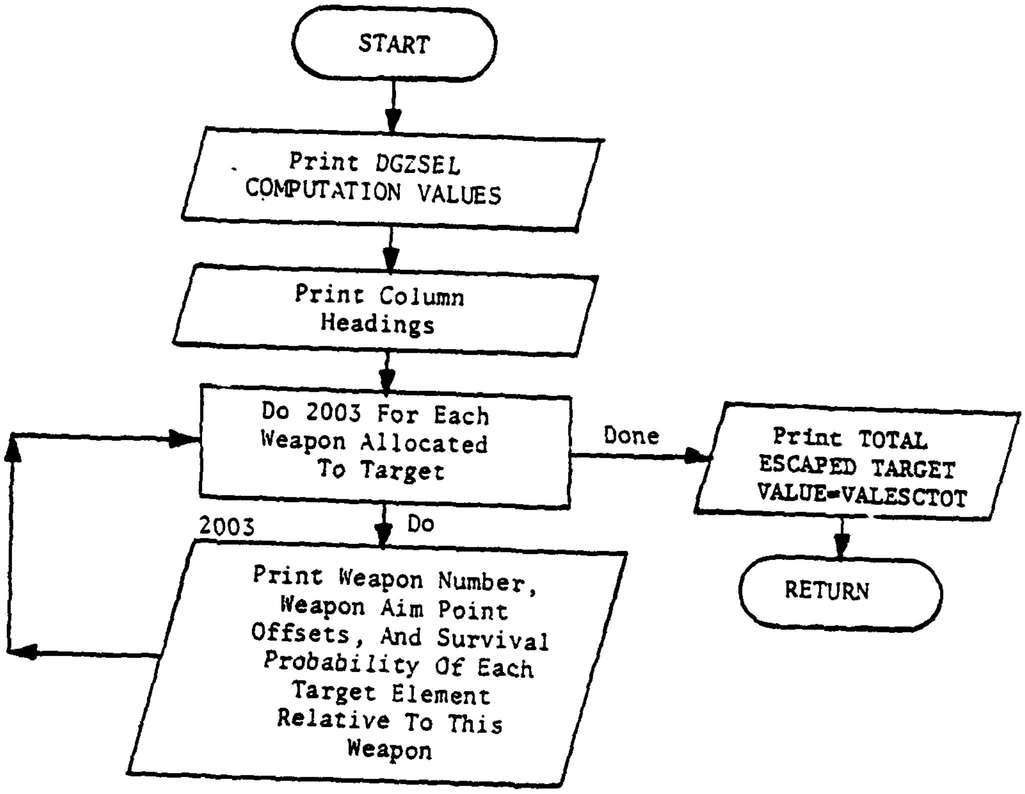

START
Print DGZSEL COMPUTATION VALUES
Print Column Headings
Do 2003 For Each Weapon Allocated To Target
Done
Print TOTAL ESCAPED TARGET VALUE=VALESCTOT
RETURN
2003
Do
Print Weapon Number, Weapon Aim Point Offsets, And Survival Probability Of Each Target Element Relative To This Weapon

Figure 80. Subroutine SEECALC

5.7.11.1 Subroutine SEEINPUT

PURPOSE: To print target data input to DGZSEL when so requested by ONPRINT 5.

ENTRY POINTS: SEEINPUT

FORMAL PARAMETERS: None

COMMON BLOCKS: C1, IONPRT

SUBROUTINES CALLED: None

CALLED BY: DGZ

Method:

If ONPRINTS 5 was specified by the user, SEEINPUT prints the title TARGET DATA INPUT TO DGZSEL DEBUG PRINT and various column headers. Subsequently, it prints data from [illegible]N C1.

Subroutine SEEINPUT is illustrated in figure 80.1.

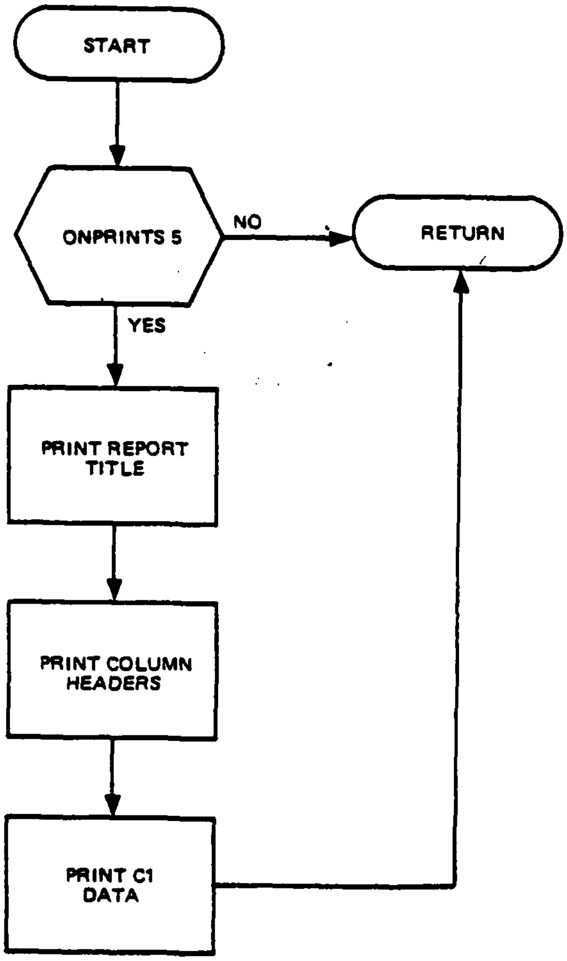


Figure 80.1. Subroutine SEEINPUT

5.7.12 Subroutine VAL

PURPOSE: VAL determines the target value which has escaped for a given weapon configuration and also determines the effective value, F_{ji}, for each target element as seen by each weapon.

ENTRY POINTS: VAL

FORMAL PARAMETERS: VESCTOT

COMMON BLOCKS: C1

SUBROUTINES CALLED: None

CALLED BY: DGZ, F2BMIN

Method:

This computation uses the effective values, VEFF(J,I), the survival probabilities, S(J,I), and the time dependent target values.

Subroutine VAL is illustrated in figure 81.

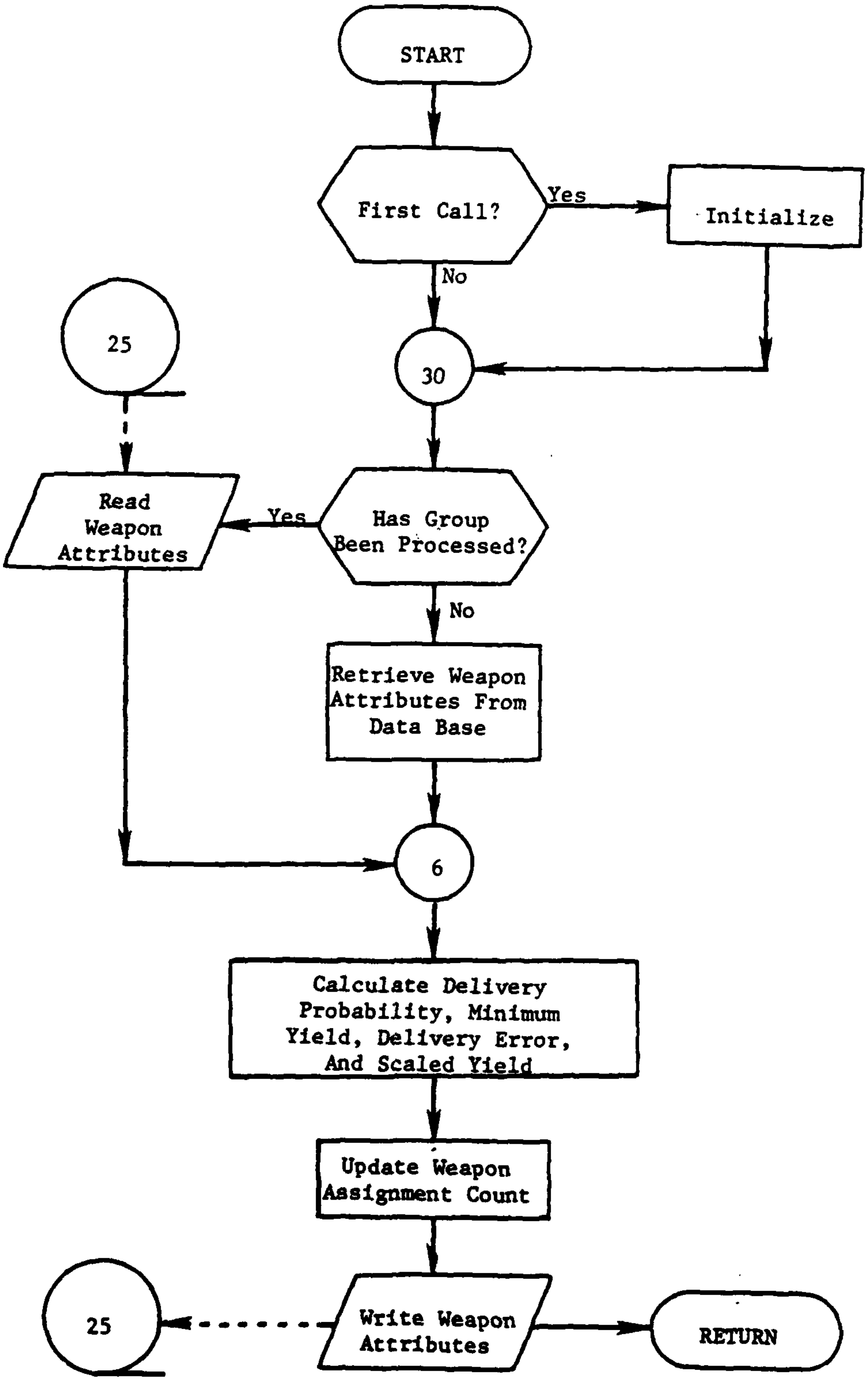

START
First Call?
Yes
Initialize
No
25
30
Read Weapon Attributes
Yes
Has Group Been Processed?
No
Retrieve Weapon Attributes From Data Base
6
Calculate Delivery Probability, Minimum Yield, Delivery Error, And Scaled Yield
Update Weapon Assignment Count
25
Write Weapon Attributes
RETURN

Figure 83. Subroutine WEPGET

5.8 Subroutine SUMPRN

PURPOSE:	To sort weapon group assignment chains and print out assignment summaries
ENTRY POINTS:	SUMPRN
FORMAL PARAMETERS:	None
COMMON BLOCKS:	C10, C30, GRPY, IONPRT, JAZ
SUBROUTINES CALLED:	DIRECT, DLETE, HEAD, NEXTTT, ORDER, REORDER, SORTIT, TIMEME
CALLED BY:	ENTMOD

Method:

This subroutine replaces existing ASSIGN records which are in no particular order on the MYASGN chain with the same set of ASSIGN records in the following sorts: For missile groups, assignments are sorted on salvo number ascending, and, within salvo, on the value of the RVAL attribute descending. For bomber groups, assignments are sorted on corridor with the corridor most often assigned occurring first, and, within corridor, on the value of the RVAL attribute descending.

The method used is to cycle the weapon group chain and perform essentially the same process for each group. First a record is read from random access file 25. This record contains counts of the weapon groups assignments. For missile groups the total number is in CORCNT(1) and -1 in CORCNT(2). For bombers the contents of CORCNT(I) corresponds to the number of assignments to corridor I. At this point, if the group is a missile group, the print header is produced. If the group is a bomber group, the assignments are totaled and the proper corridor order is stored in CORORD.

Now the assignments are read from the MYASGN chain and stored in array F. When the assignments have all been processed, the sort keys are generated. Then the assignments are sorted by routines ORDER and REORDER. The old assignments are deleted from the MYASGN chain. Now each assignment in sort order, is retrieved from array F. A new ASSIGN record is stored in the MYASGN chain and the assignment is printed. The bomber header is produced each time a new corridor is encountered.

Subroutine SUMPRN is illustrated in figure 84.

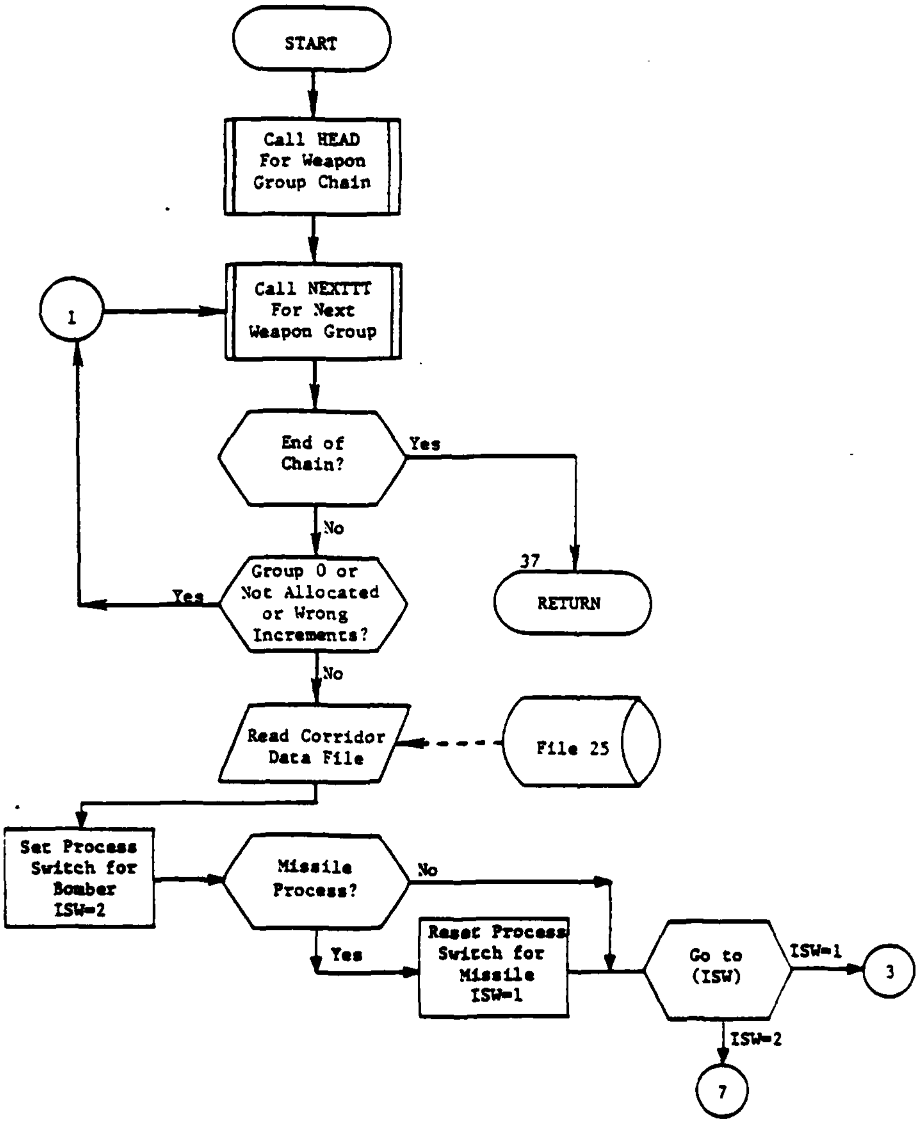

START
Call HEAD For Weapon Group Chain
1
Call NEXTTT For Next Weapon Group
End of Chain?
Yes
No
37
RETURN
Group 0 or Not Allocated or Wrong Increments?
Yes
No
Read Corridor Data File
File 25
Set Process Switch for Bomber ISW=2
Missile Process?
No
Yes
Reset Process Switch for Missile ISW=1
Go to (ISW)
ISW=1
3
ISW=2
7

Figure 84. Subroutine SUMPRN (Part 1 of 12)

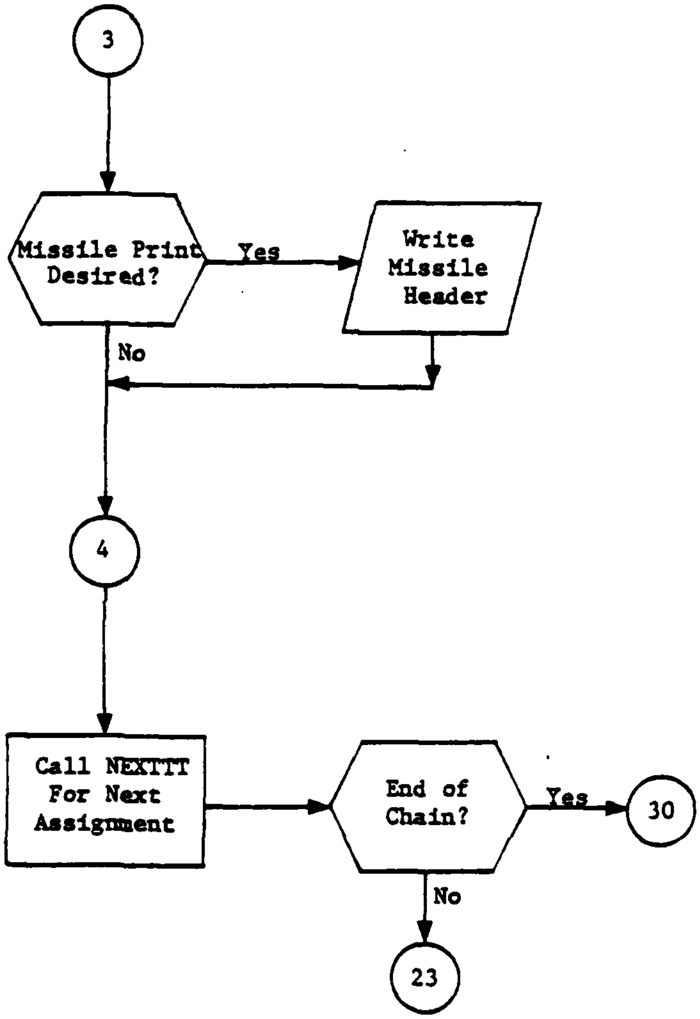


Figure 84. (Part 2 of 12)

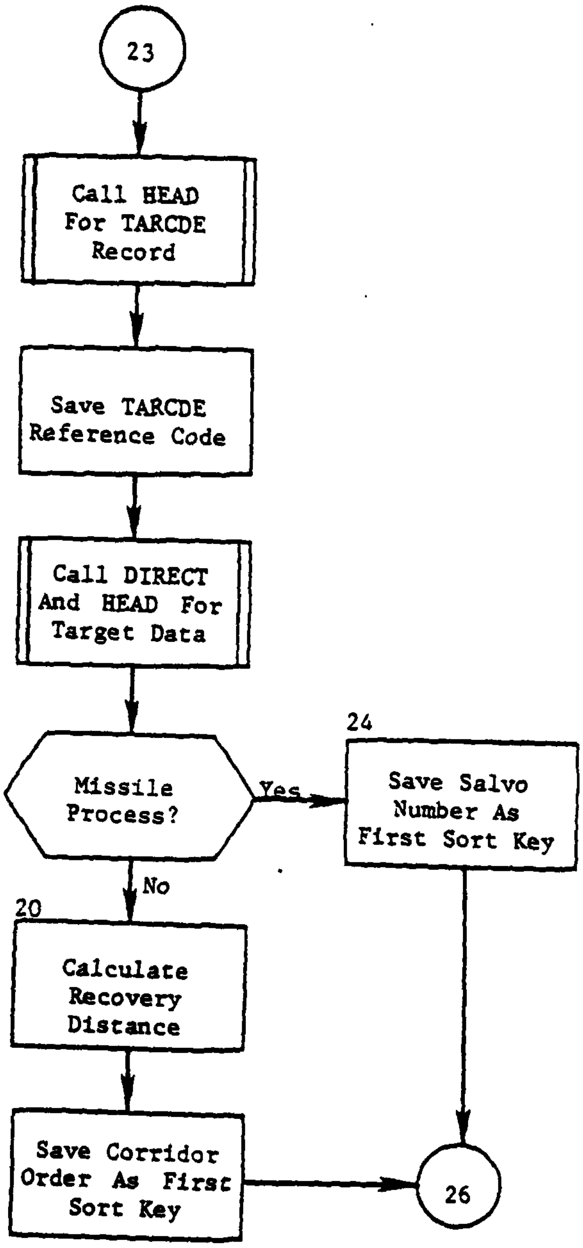

23
Call HEAD For TARCDE Record
Save TARCDE Reference Code
Call DIRECT And HEAD For Target Data
Missile Process?
Yes
24
Save Salvo Number As First Sort Key
No
20
Calculate Recovery Distance
Save Corridor Order As First Sort Key
26

Figure 84. (Part 3 *of* 12)

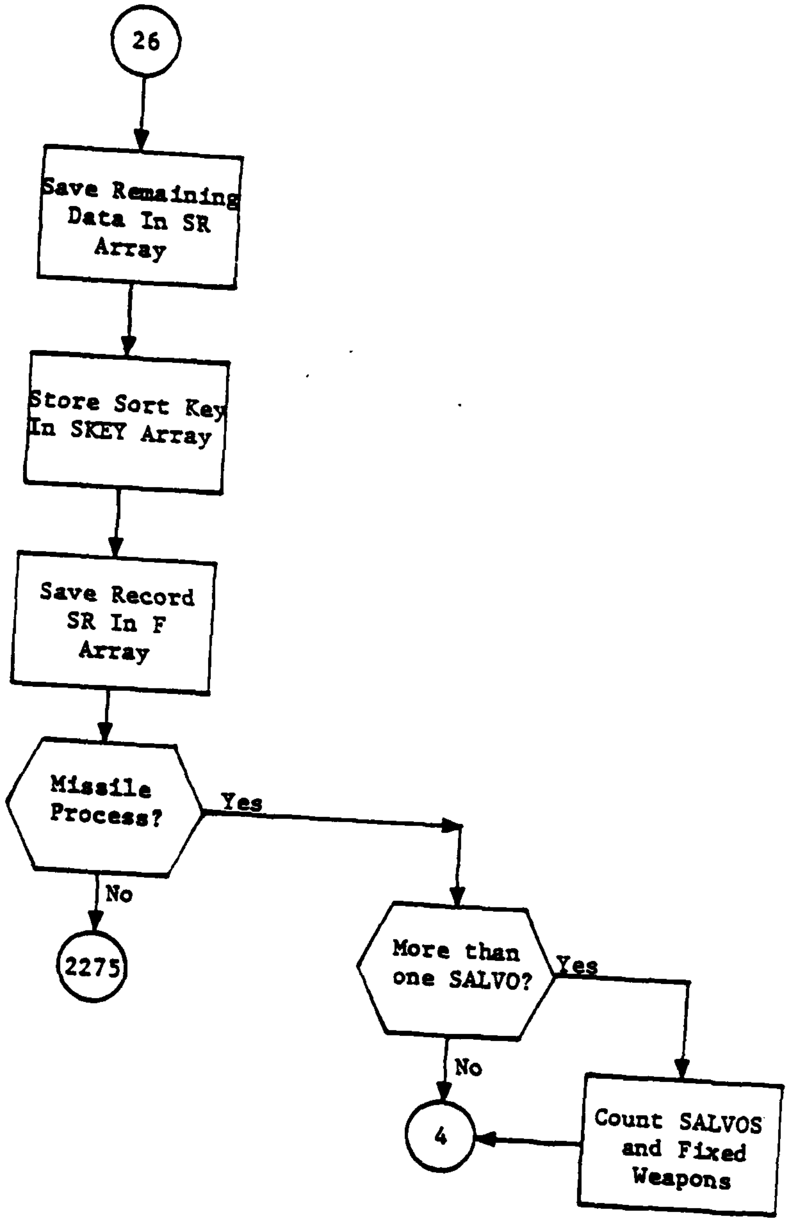

26
Save Remaining Data In SR Array
Store Sort Key In SKEY Array
Save Record SR In F Array
Missile Process?
Yes
No
2275
More than one SALVO?
Yes
No
4
Count SALVOS and Fixed Weapons

Figure 84. (Part 4 of 12)

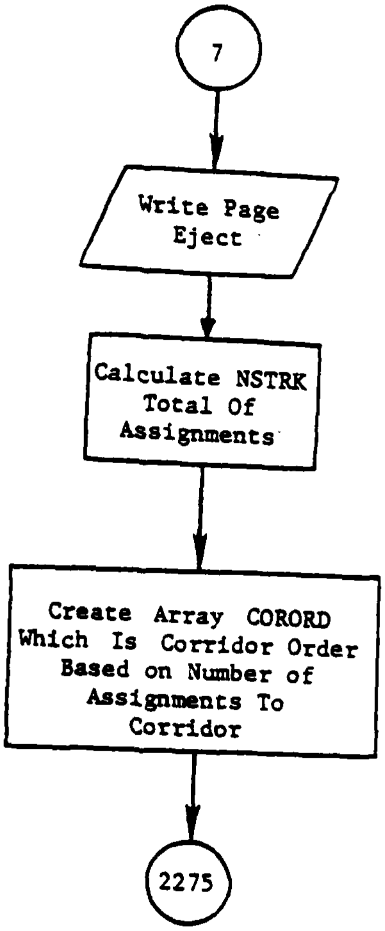

7
Write Page Eject
Calculate NSTRK Total Of Assignments
Create Array CORORD Which Is Corridor Order Based on Number of Assignments To Corridor
2275

Figure 84. (Part 5 of 12)

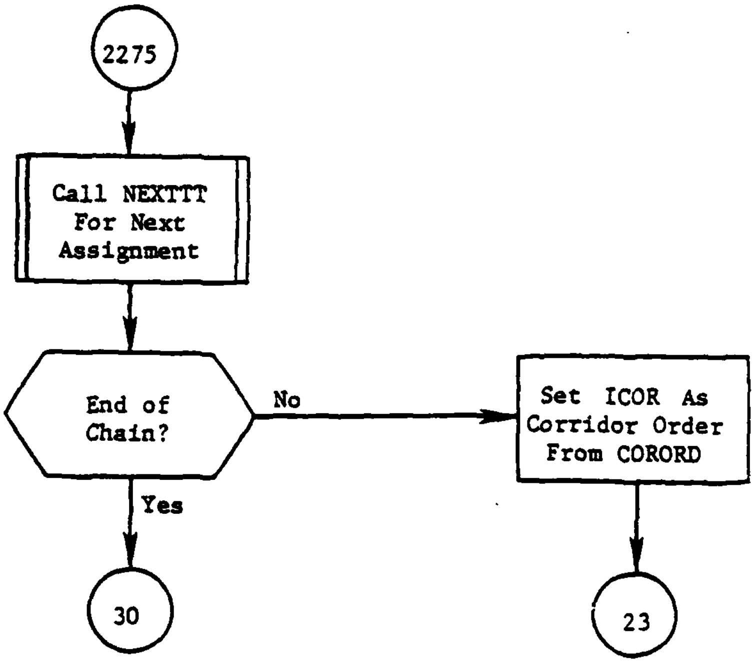

2275
Call NEXTTT For Next Assignment
End of Chain?
No
Yes
Set ICOR As Corridor Order From CORORD
30
23

Figure 84. (Part 6 of 12)

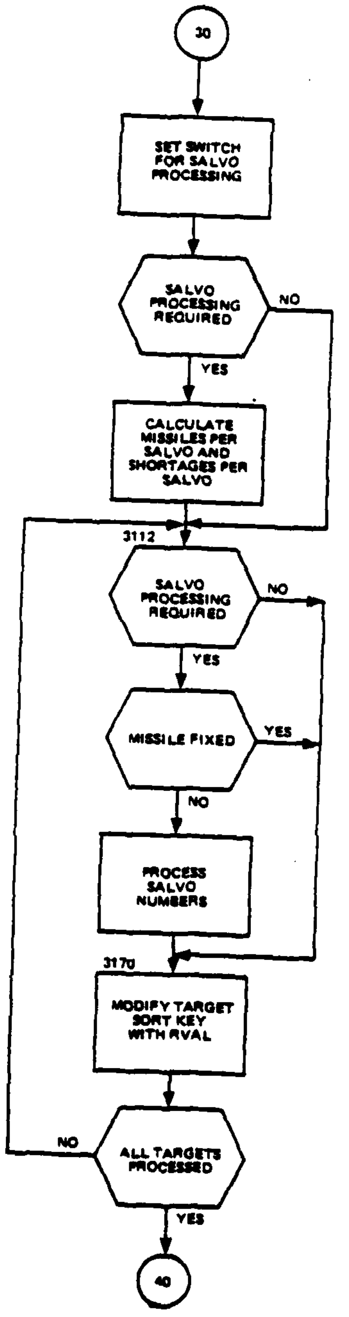

30
SET SWITCH FOR SALVO PROCESSING
SALVO PROCESSING REQUIRED
NO
YES
CALCULATE MISSILES PER SALVO AND SHORTAGES PER SALVO
3112
SALVO PROCESSING REQUIRED
NO
YES
MISSILE FIXED
YES
NO
PROCESS SALVO NUMBERS
3170
MODIFY TARGET SORT KEY WITH RVAL
NO
ALL TARGETS PROCESSED
YES
40

Figure 84. (Part 7 of 12)

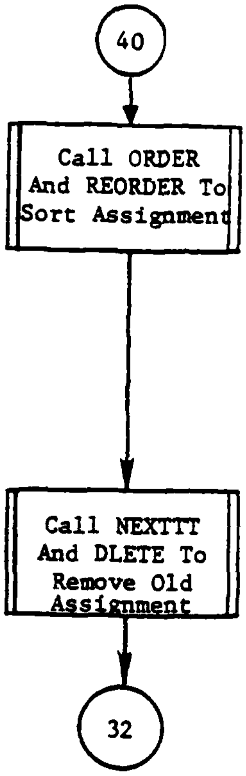

40
Call ORDER And REORDER To Sort Assignment
Call NEXTTT And DLETE To Remove Old Assignment
32

Figure 84. (Part 8 of 12)

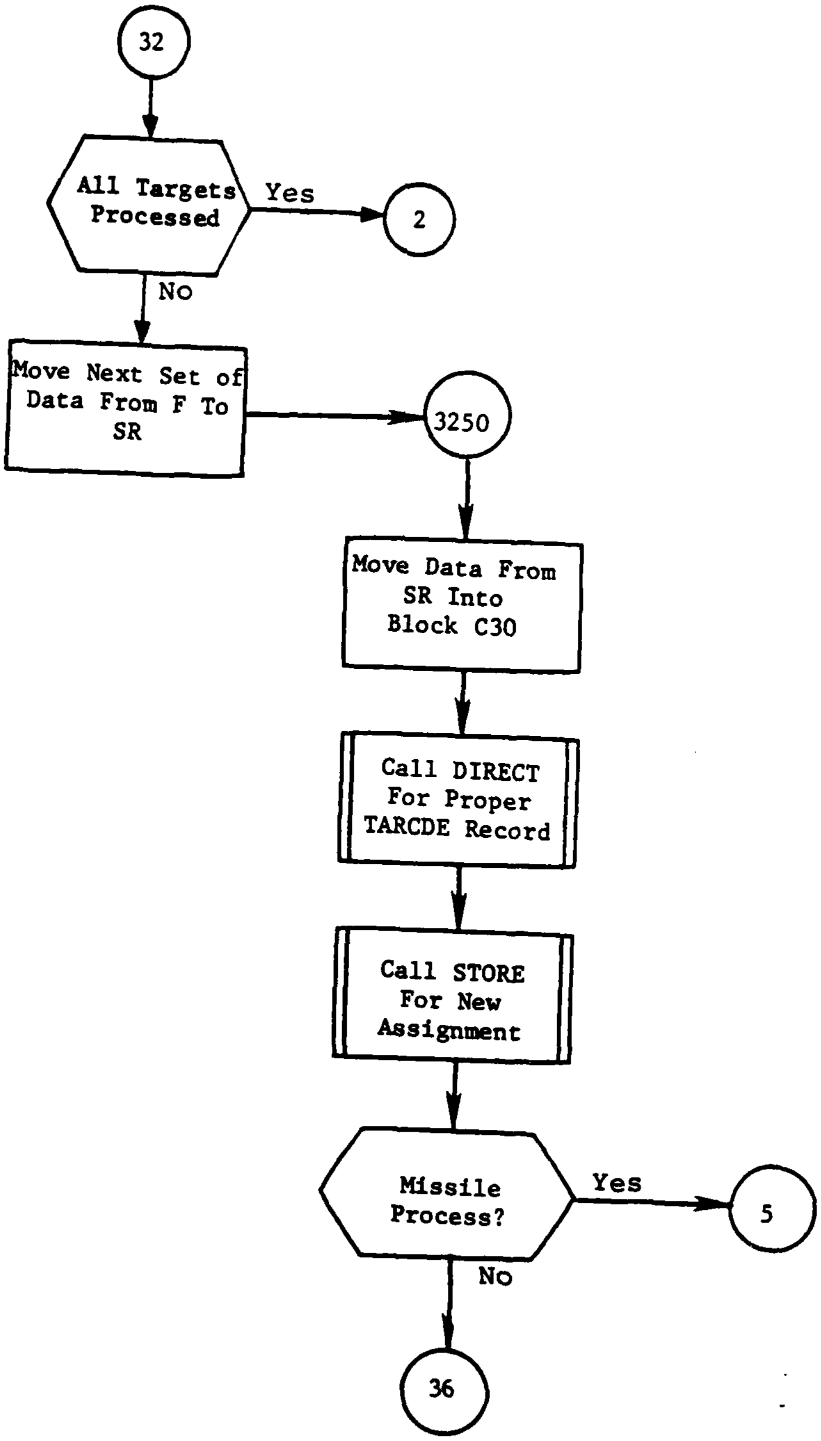

32
All Targets Processed
Yes
2
No
Move Next Set of Data From F To SR
3250
Move Data From SR Into Block C30
Call DIRECT For Proper TARCDE Record
Call STORE For New Assignment
Missile Process?
Yes
5
No
36

Figure 84. (Part 7 of 9)

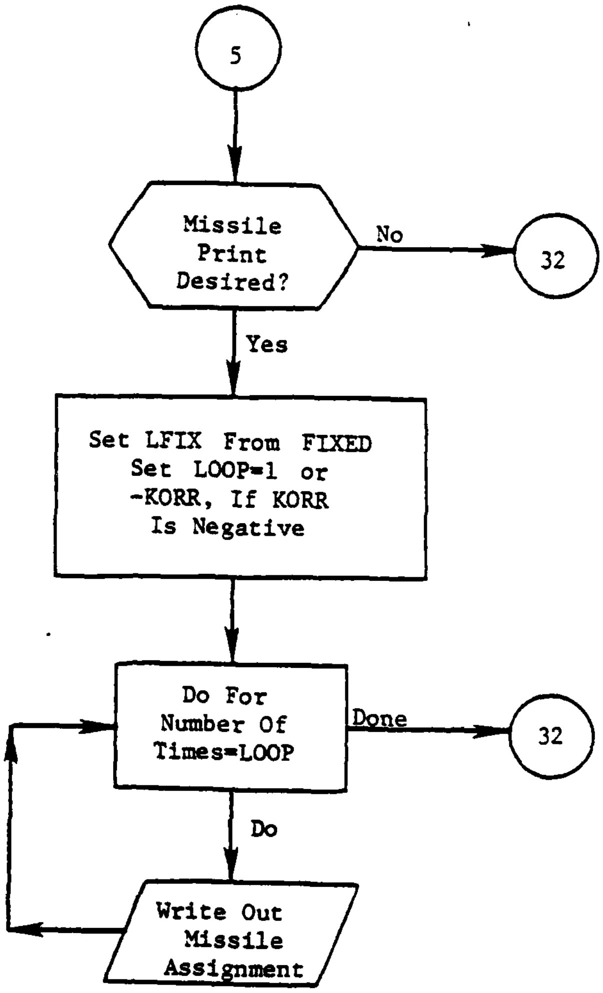

5
Missile Print Desired?
No
32
Yes
Set LFIX From FIXED Set LOOP=1 or -KORR, If KORR Is Negative
Do For Number Of Times=LOOP
Done
32
Do
Write Out Missile Assignment

Figure 84. (Part 10 of 12)

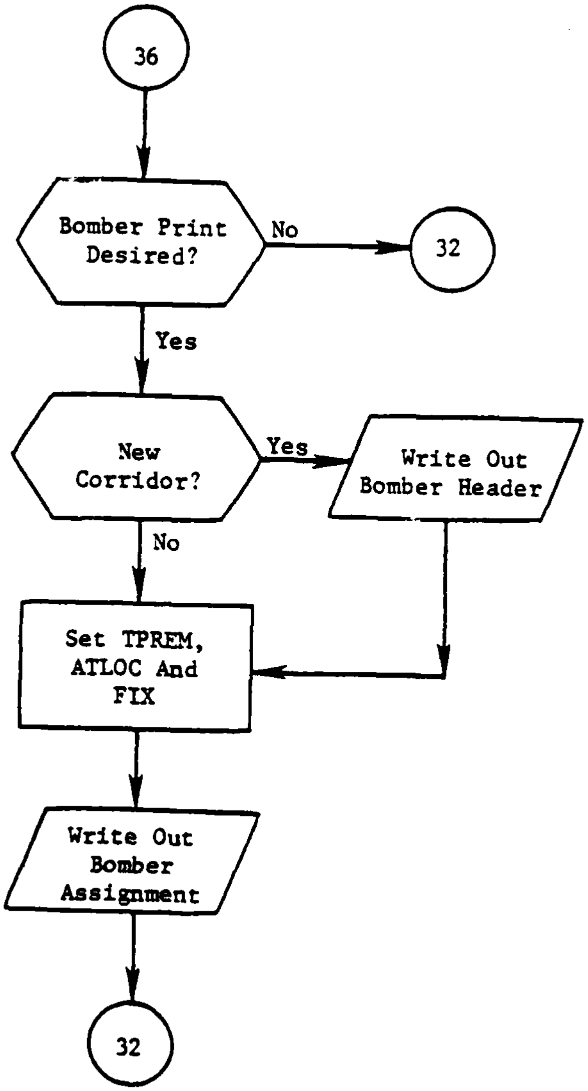


Figure 84. (Part 11 of 12)

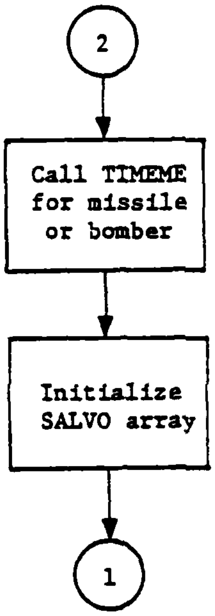


Figure 84. (Part 12 of 12)

5.9 Subroutine MINIOUT*

PURPOSE:	Produce miniallocator STRIKE tape
ENTRY POINTS:	MINIOUT
FORMAL PARAMETERS:	None
COMMON BLOCKS:	C10, C15, C30, GAMETIME, IONPRT
SUBROUTINES CALLED:	ABORT, CONVLL, DISTF, FINDTIME, HDFND, HEAD, IGETHOB, INFORM, IPROB, NEXTTT, RETRV
CALLED BY:	ENTMOD (ALOCOUT)

Method:

This subroutine first retrieves the necessary weapon, payload and warhead data. Then each group is processed in order. For each group, every associated weapon assignment record is used to develop and reformat the necessary data to produce a STRIKE tape record (for STRIKE tape format consult CSM MM 9-77 Volume IV). This tape is thus produced.

Subroutine MINIOUT is illustrated in figure 84.1.

* Main subroutine of overlay MINIO.

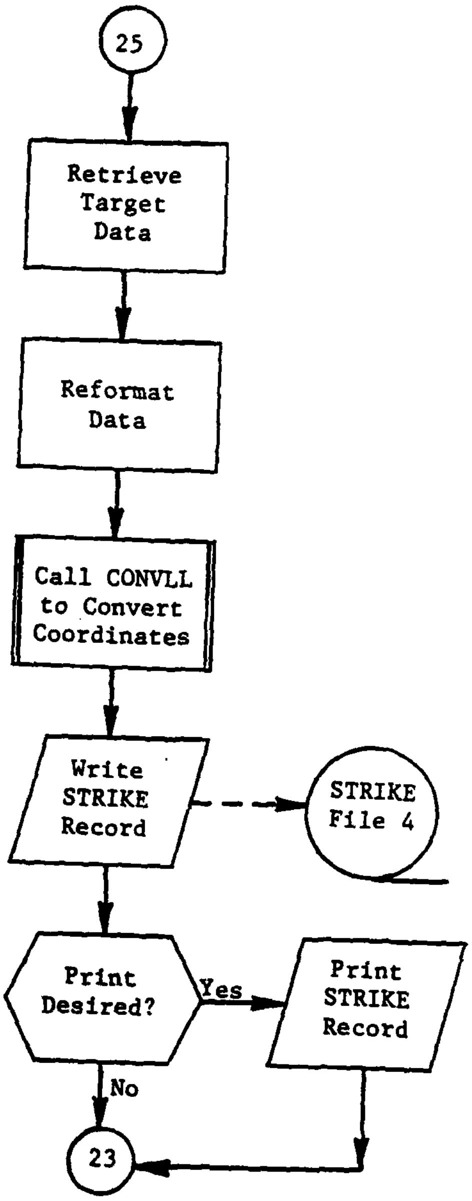


Figure 84.1. (Part 3 of 3)

5.9.2 Subroutine FINDTIME

PURPOSE: Calculate and pack strike time

ENTRY POINTS: FINDTIME

FORMAL PARAMETERS: XX - input strike time
II - output strike time (packed)

COMMON BLOCKS: GAMETIME

SUBROUTINES CALLED: None

CALLED BY: MINIOUT

Method:

Day and month are set from the /GAMETIME/ block. HHR is added to the input time and hours, minutes and seconds calculated. The results are packed into II and returned.

Subroutine FINDTIME is illustrated in figure 84.3.

PRECEDING PAGE BLANK-NOT FILMED

$$\mathrm{PINT}=\mathrm{PKTX}*[(\mathrm{PXLOW}*\mathrm{RXLOW})+(\mathrm{PXHIGH}*\mathrm{RXHIGH})+(1-\mathrm{PXLOW}-\mathrm{PXHIGH})]*\mathrm{MISDEF}$$

This variable is the expected number of objects to be removed by the terminal defense interceptors.

The penetration probability for any warhead is defined as

$$1.0 - \left[\frac{\mathrm{PINT}}{\mathrm{NOBJ}}\right]$$

If this probability is less than (1.0 - PKTX) it is reset to that value.

A.2 Weapon/Target Interaction

The quality of the plans, in terms of realism and sophistication will be a direct reflection of the realism incorporated in the payoff function. In order to produce plans of maximum realism, the payoff function should reflect all the major factors that would be considered by an experienced military planner. The design incorporates:

1. Time of arrival of weapons

2. Time dependence of target values, which can reflect a planner's uncertainty in the time of arrival of weapons relative to change in target value

3. Weapon range limitations

4. Uncertainty in target vulnerability

5. Correlations in the effectiveness of weapons of similar nature reflecting such factors as reliability, DBL probability, and defense effectiveness.

To evaluate the capability of any weapon group against any target, program ALOC requires six basic numbers. These are:

SBL(G) The probability assumed that weapons in group G are not destroyed before launch

CC(KR) The assumed command and control reliability associated with the region for group G

REL(K) The assumed reliability for weapon type K used by group G

PEN(G) The estimated penetration probability for weapons from group G to the target

STK(G,JH) The estimated kill probability of warheads in group G if delivered against the JH hardness component of the target

TVALTOA(G) The estimated target value at the time of weapon arrival for weapon from group G (this factor is computed from the time of arrival for a weapon from group G, TOA[G]).

These numbers reflect the planning factors the user has specified for the plan generation and do not necessarily reflect the values that the user specifies for external simulation. The number is noted as "assumed" where it is a direct user input supplied in the data base. It is described as "estimated" where it is a derived quantity, based on other input data.

Actually, the numbers reflect only two types of information -- the time of arrival information, and the kill probability data. The single shot kill probability is simply a product of the first five items. The breakdown of the single shot kill probability into these five separate factors, however, is required in order to estimate correlations in delivery probability between several warheads delivered to the same target.

Most of the processing of weapon/target interactions deals with the six quantities given above. These quantities are then used in the calculation of weapon payoff.

The basic payoff calculation is modified by the inclusion of weapon correlation considerations. For each single weapon, four factors are calculated: the single shot kill probability and three auxiliary quantities required by the correlation model (see Weapon Correlations, this appendix).

If we define the overall single shot kill probability on one hardness
component J as: $SSK = REL * CC * SBL * PEN * STK$
then $MUP(G,J) = - LOGF(1.0 - SSK)$
and $SSIG(G,J) - MUP(G,J)/-LOGF(SSK)$

If the option to use the square root damage law is selected, MUP is defined in a different manner. It is defined so that:

$$(1.0 - SSK) = \left(1 + \sqrt{MUP(G,J)}\right) *\exp \left(- \sqrt{MUP(G,J)}\right)$$

The use of the square root damage function is further explained in a later section (see Multiple Weapon Attacks -- Square Root Law, this appendix).

MUP is in effect a measure of the effectiveness of the weapon against the specified hardness component. If all weapons were independent, the survival probability for the component with respect to multiple weapons IG would be simply:

$$EXPF -\left(\Sigma\ MUP(IG,J)\right)$$

(This is called the exponential damage law.)

$$P = .1/(\text{MAXSAL} - 1)$$

where MAXSAL is the maximum salvo number in the group and .1 is a sensitivity parameter to slow the rate of change of P.

Closing Factors -- Premiums: The Lagrange multiplier for each weapon is modified by a premium. This factor is used to force closure of weapon allocations to the available stockpile. It acts as a bonus for using under-allocated weapons and a penalty for using overallocated weapons. The parameters which are used to calculate the premiums are:

SURPWP(G) An estimate of the number of surplus (or unallocated weapons) in the group. This number is based on estimated allocation rates in the early phase and the actual allocation later.

NWPNS(G) The actual number of weapons in group G.

LAMEF(G) The Lagrange multiplier for the group.

The premium depends also on three control parameters: PROGRESS, PRM, and CLOSE.*

The effect of PROGRESS (described earlier) is as follows:

1. If PROGRESS is greater than 1.0, this indicates that a verification allocation is desired to obtain a theoretical upper bound on the payoff without regard to meeting the actual stockpile constraints. For this purpose, the premiums are simply set to zero.

2. If PROGRESS is less than 1.0, a small premium is computed which is intended only to avoid large deviations from the desired allocation rate of small errors in the Lagrange multipliers.

   (Otherwise, a trivial change in the multipliers for two competing weapons could result in a complete change from always allocating one to always allocating the other.)

3. If PROGRESS is equal to 1.0, this is a signal that the closing phase has been reached and the object is to close in on an exact allocation of the available weapons. In this case, a larger step function premium is computed, and the size of the step function is gradually increased until final closure occurs.

* PROGRESS is set internally by the module as described in Section 3.

During the early allocation phase, superimposed on the actual payoff is a small negative quantity (called a premium) that is proportional to the value of each weapon group and quadratic in the size of the error in allocation. In effect, the actual payoff, H(X), for any allocation, X is adjusted to $H(X)_{ADJ}$:

$$H(X)-PRM*\sum_{G}\left\{NWPNS(G)*LAMEF(G)*\left(\frac{SURPWP(G)}{NWPNS(G)}\right)^{2}\right\}$$

This quadratic addition to the payoff function has the effect of introducing a preference for allocations where the absolute value of SURPWP is small.

The addition or deletion of a weapon from group G will give rise to a difference in SURPWP equal to the current target multiplicity. Thus, the change in this quantity (per unit multiplicity) with the addition of a weapon G is:

$$PREMIUM(G)=PRM*LAMEF(G)*\frac{SURPWP(G)\ -\ .5}{NWPNS(G)}$$

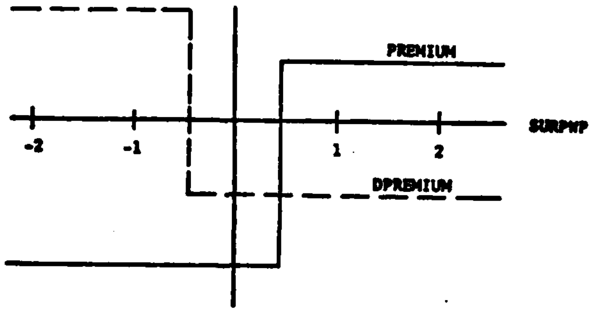
and the change with deletion of a weapon is:

$$DPREMIUM(G)=PRM*LAMEF(G)*\frac{-SURPWP(G)\ -\ .5}{NWPNS(G)}$$

The value of PRM is a user-input parameter. The value should be less than 1.0. Otherwise, in cases when no weapons from some group have been used, the premium for allocation of a weapon could exceed the cost of the weapon LAMEF(G) and weapons could be allocated even if the payoff were zero or even negative. Experience has shown that values between .5 and .9 work very well.

When PROGRESS reaches 1.0, PRM is set to .9 by the program to accelerate convergence. In addition, a small step function is added.

The following sketch illustrates the value of these step function premiums as a function of their SURPWP:



EXPASM = fraction of weapons in a group which are ASMs

EXPBMB = 1 -EXPASM = fraction of weapons which are bombs

DEA = expected destroyed value of target if ASM used (calculated in program ALOC)

DEB = expected destroyed value of target if bomb used (calculated in program ALOC)

AVDE = average (by group) of quantity ABSF (DEA -DEB)

FASM = current fraction of weapons allocated which are ASMs (calculated in program ALOC)

FBOMB = 1 -FASM = current fraction of weapons allocated which are bombs

CONPAY = internal program variable between 0. and 1.

Except for CONPAY, all of the above variables are defined for each group composed of bombers. For each bomber group on each target the allocation process selects the type of warhead (ASM or bomb) which is to be used on the target. When the value of PROGRESS is zero or two, the preferred weapon is the weapon with the higher DE (i.e., ASM will be selected if DEA is greater than DEB and <u>vice versa</u>). For values of PROGRESS of .4, .5, .75, and 1.0, the selection process will consider the allocation franctions of the ASMs and bombs, as described in the following paragraphs.

If ASMs are underallocated, ASMs are selected as the preferred weapon <u>unless</u> DEB is greater than DEA <u>and</u> the quantity (EXPASM -FASM)/EXPASM is less than or equal to the quantity CONPAY*(DEB-DEA)/AVDE. If both of these two conditions are met, then the preferred weapon is the bomb.

Note that the quantity (EXPASM -FASM)/EXPASM provides a measure of the size of the allocation imbalance. If ASMs are only slightly underallocated, this quantity will be very small (near zero). If there is a great difference between the actual and allocated fractions, this quantity will approach the value one. The quantity (DEB-DEA)/AVDE is a measure of the magnitude of the damage difference relative to the average damage difference. The quantity ranges from a low of zero to high positive values. A value of one for the quantity represents an average damage difference. The variable CONPAY is used to reflect the importance of the allocation difference relative to the damage difference. Thus, the conditions of the preceding paragraph select the bomb as the preferred weapon if the allocation difference is less than the modified damage difference. Thus, if the allocation is nearly correct, the more damaging weapon is likely to be chosen as preferred. If the allocation is far from correct, the underallocated weapon will be selected on all targets except those where the damage difference is quite large. The same rationale holds for the case of underallocated bombs as described in the next paragraph.

If bombs are underallocated, bombs are selected as the preferred weapon unless DEA is greater than DEB and the quantity (EXPBMB -FBOMB)/EXPBMB is less than or equal to the quantity CONPAY*(DEA -DEB)/AVDE. If both these conditions are met, then the preferred weapon is the ASM.

The value of variable CONPAY lies between zero and one. Lower values of CONPAY tend to increase the importance of the allocation difference. High values of CONPAY increase the importance of the damage difference. In order to provide adequate closing force, the value of CONPAY decreases as the value of PROGRESS increases. Additionally, when PROGRESS equals one, the value of CONPAY continues to decrease.

The selection of ASM or bomb on a particular target allows the allocation process to assess correctly the expected damage effects. Bombs and ASMs usually differ greatly in yield, penetration probability, CEP, and delivery probability. By differentiating between these weapon types at allocation time, the allocation program selects the best weapon to be used when the bomber sorties are generated. The balance between allocation and damage differences provides for maximization of damage while continuing consideration of actual weapon stockpiles.

A.5 Derivation of Lagrange Multiplier Adjustment

Define the following variables:

CURSUM(J) = sum of the target weights from last Lagrange multiplier update

NOWPS(J) = number of weapons sharing attribute J

NTGTS = number of targets

SNSTVTY
FSNSTVTY } = user-input parameters which control rate of multiplier adjustment

LAMEF(G) = Lagrange multiplier for group G

LA(J) = Lagrange multiplier for attribute J; J = ALL, CLASS, TYPE, etc.

PRM = local internal control variable which governs size of premiums (closing factors)

NWPNS(G) = number of weapons in group G

If we now associate the variable PARTIAL with n this gives rise to the following procedure for updating LA:

$$LA_1(J) = LA_0(J)* [1.0 + \frac{CORFAC*ALERREST(J, INTPRD)/(- PARTIAL)}{ALERREST(J, INTPRD) + (NOWPS(J)/NTGTS)}]$$

ALERREST(J) is computed as

$$ALERREST(J) = \frac{RUNSUM(J)}{WTSUM(J)} - \frac{NOWPS(J)}{NTGTS}$$

The formula for $LA_1(J)$ is well-behaved if ALERREST is large and positive but if it is negative and as large as the expected rate (NOWPS(J)/NTGTS) (i.e., if the actual allocation rate is zero), then the denominator goes to zero. In this case an infinite correction would be indicated. To avoid this, the expected rate in the denominator is multiplied by 2 giving:

$$LA_1(J) = LA_0(J)* [1.0 + \frac{CORFAC*ALERREST(J, INTPRD)/(-PARTIAL)}{ALERREST(J, INTPRD) + 2*(NOWPS(J)/NTGTS)}]$$

This is the function used. The new Lambda's $LA_1(J)$, are recomputed for attribute J (e.g., Jall, Jclass) and for every MULSTEP targets as previously outlined.

In the present version of the program the value of PARTIAL(J) has been set equal to 1.0 for all the local multipliers LA(J). This choice is based on the effect of the return on the sensitivity of the allocation rate to the value of LAMEF or λ. When the multipliers are almost correct, it is usually the case that most weapon groups are in close competition with many other groups with very similar properties. Then a small change in the multiplier LAMEF will produce a very large change in the allocation rates, as the weapon group in question almost totally replaces, or is replaced by, its competitors.

However, such a large error in the allocation rate will not actually occur because as the error builds up the estimated value of the payoff will be automatically changed by the premium. Thus for constant values of LAMEF, when an equilibrium allocation rate is reached, it must be approximately true that the error in LAMEF is compensated by the premium. This is, if λ_0 is the correct value for LAMEF then:

$$LAMEF - PREMIUM \cong \lambda_0$$

Since:

$$PREMIUM = PRM*LAMEF* \frac{SURPWP -.5}{NWPNS}$$

we can define a relation between LAMEF and (SURPWP/NWPNS)

$$\text{LAMEF}*(1 - \text{PRM}* \frac{\text{SURPWP} -.5}{\text{NWPNS}}) \cong \lambda_0$$

Since this relationship is the same for all groups it is reasonable simply to use the same value 1.0 of partial derivative for all local multipliers.

The values of LAMEF(G), where G is the group index, are recomputed using the new values of the local multipliers (LA(J) accordingly,

$$\text{LAMEF(G)} = \text{LA(Jall)}*\text{LA(Jclass)}*\text{LA(Jreg)}*\text{LA(Jalert)}*\text{LA(Jgroup)}$$

At the same time it is necessary to reevaluate the summation of the value of all the weapons VALWPNS = Σ LAMEF(G)*NWPNS(G) and the summation of the value of the error in weapons allocated

$$\text{VALERR} = \Sigma\ \text{LAMEF(G)}*\text{ABSF(SURPWP(G))}$$

using the updated values of LAMEF.

Target Weight Change Rate & Integration Period: The above explained how multipliers are recomputed by monitoring allocation rates. The remaining discussion addresses how the target weight change rate and integration period is computed.

The average number of targets over which allocation rates are averaged (the integration period) is determined by the rate at which the target weights are increased.

In estimating the rate with which to correct multipliers, it was computed on a statistical basis that even if the allocation rates were correct an estimated error of size ALERREST would be expected if the allocation rates were monitored only over a small sample of M targets where:

$$\text{M} = (\text{NOWPS(J)}/\text{NTGTS})/(\text{ALERREST(J)})^2$$

Thus if separate integration periods could be used for each local multiplier, M as defined above might provide a reasonable basis for determining the period. However, in fact, the same periods must be used for all local multipliers LA(J). Currently three periods are maintained (INTPRD=1, 2, 3). Consequently the value of the integration period used must be based on an estimate of overall error rate. The corresponding relation is:

$$\text{M} = (\sum_{G} \text{NOWPS(J)}/\text{NTGTS})/\sum_{G} (\text{ALERREST(J)})^2$$

where the summations are taken over all weapon groups. The quantity $\sum_{G}$NOWPS(J), is identical with NOWPS(1) (Note: LA(J) for J = 1 is used

From the previous equations,

$$\bar{V}_i = \left[\prod_{k=1}^{i} S_k\right] V(T_i) \quad \text{and} \quad E_i = \left[\prod_{k=1}^{i} S_k\right]\left[V(T_{i-1}) - V(T_i)\right]$$

(For i = 1, the product $\left(\prod_{k=1}^{i-1} S_k\right)$ is understood = 1. Also $V(T_{N+1}) = 0$.)

The total escaping value associated with target j is

$$\sum_{i=1}^{N+1} E_{ij} = \sum_{i=1}^{N+1}\left(\left[\prod_{k=1}^{i-1} S_{kj}\right]\left[V_j(T_{i-1}) - V_j(T_i)\right]\right)$$

The value on target j which escapes after arrival of weapon i is given by

$$\sum_{p=i+1}^{N+1} E_{pj}$$

The effective value of target j associated with weapon i defined by

$$F_{ij} = \left(\sum_{p=i+1}^{N+1} E_{pj}\right) \Big/ S_{ij}$$

This value is introduced for computational efficiency and may be thought of as the total value available for weapon i, the effect of all other weapons having been taken into account.

The marginal value picked up on target j due to weapon i is given by

$$F_{ij}(1 - S_{ij})$$

where S_{ij} is a function of, among other things, the position of weapon i. For a fixed weapon configuration, weapon i can be moved from (x,y) to (x',y') and the <u>marginal escaped value</u> is given by:

$$\sum_{j=1}^{NT} F_{ij}(S_{ij} - S'_{ij})$$

To establish an initial weapon configuration, a lay-down is performed as follows. Initially, set $S_{ij} = 1$ for all i, j. Denote by S^j_{ik} the survival probability of the kth target, relative to the ith weapon, when this weapon is placed on the jth target. Now the ith weapon is placed on that target, j, which yields a maximum value for the expression

$$\sum_{k=1}^{NT} F_{ik}(S_{ik} - S^j_{ik})$$

The S_{ik} are now set to equal to S^j_{ik} (k = 1,2,..., NT) the F_{ik} (all i, k) are redetermined, i is increased by one, and the process repeated until all weapons have been allocated.

This weapon configuration can now be input as the initial position to a "hill climber" routine, based on a steepest descent algorithm, which attempts to optimize further by replacing the discrete set of possible weapon positions with the two-dimensional continuum. The function to be minimized is:

$$\sum_{j=1}^{NT} \sum_{i=1}^{N+1} E_{ij}$$

Processing by the optimizer will be terminated either when the optimum has been achieved or when a specified number of iterations have been completed. In either case, to insure that the local optimum obtained cannot be further improved, the value of removing, in sequence, each of the weapons from its final location and placing it on one of the target points is explored. If the results obtained by this method are better than those achieved with the previous configuration, this new assignment will be used as an initial one for a second utilization of subroutine FINDMIN. If not, the results of the first use of subroutine FINDMIN will be kept.

DISTRIBUTION

Addressee	Copies
CCTC Codes	
C124 (Reference and Record Set)	3
C124 (Stock)	6
C126	2
C313	1
C314	7
C630	1
DCA Code	
205	1
EXTERNAL	
Chief, Studies, Analysis and Gaming Agency, OJCS ATTN: SFD, Room 1D935, Pentagon, Washington, DC 20301	2
Chief of Naval Operations, ATTN: OP-654C, Room BE781 Pentagon, Washington, DC 20350	2
Commander-in-Chief, North American Air Defense Command ATTN: NPXYA, Ent Air Force Base, CO 80912	2
U.S. Air Force Weapons Laboratory (AFSC) ATTN: AFWL/SUL (Technical Library), Kirtland Air Force Base, NM 87117	1
Director, Strategic Target Planning, ATTN: (JPS), Offutt Air Force Base, NE 68113	2
Defense Technical Information Center, Cameron Station, Alexandria, VA 22314	12
	42